International Journal of Data Science and Analytics (2020) 10:331–347
https://doi.org/10.1007/s41060-020-00226-0

REGULAR PAPER

Streaming statistical models via Merge & Reduce

Leo N. Geppert[1] · Katja Ickstadt[1] · Alexander Munteanu[1,2] · Christian Sohler[3]

Received: 9 August 2019 / Accepted: 25 May 2020 / Published online: 12 June 2020

Abstract
Merge & Reduce is a general algorithmic scheme in the theory of data structures. Its main purpose is to transform static data structures—that support only queries—into dynamic data structures—that allow insertions of new elements—with as little overhead as possible. This can be used to turn classic offline algorithms for summarizing and analyzing data into streaming algorithms. We transfer these ideas to the setting of statistical data analysis in streaming environments. Our approach is conceptually different from previous settings where Merge & Reduce has been employed. Instead of summarizing the data, we combine the Merge & Reduce framework directly with statistical models. This enables performing computationally demanding data analysis tasks on massive data sets. The computations are divided into small tractable batches whose size is independent of the total number of observations n. The results are combined in a structured way at the cost of a bounded $O(\log n)$ factor in their memory requirements. It is only necessary, though nontrivial, to choose an appropriate statistical model and design *merge* and *reduce* operations on a casewise basis for the specific type of model. We illustrate our Merge & Reduce schemes on simulated and real-world data employing (Bayesian) linear regression models, Gaussian mixture models and generalized linear models.

Keywords Mergeable statistical models · Large data · Streaming · Distributed · Regression analysis

1 Introduction

In recent times, data sets with a massive number of observations have become more and more present, making scalability one of the main challenges of modern data analysis. For many statistical methods, these amounts of data lead to an enormous consumption of resources. A prominent example is linear regression, an important statistical tool in both frequentist and Bayesian settings. On very large data sets with n observations and d variables ($n \gg d$), carrying out regression analysis becomes increasingly demanding concerning running time and memory consumption making the analysis practically tedious or even impossible. This is especially true when the data do not fit into the fast internal memory but has to be read over and over again from slow external memory devices or databases, which blows up the *wall-clock time*, i.e., the actual elapsed time, by orders of magnitude.

In this paper, we propose a method called Merge & Reduce as a technique to address these scalability limitations in regression analysis. Merge & Reduce is well known in computer science and has mainly been used for transforming static data structures to dynamic data structures with little overhead [8]. This can be leveraged to design streaming algorithms for a computational problem based on coresets. A coreset is a significantly reduced data set which can be used instead of the full data set; the same algorithm can run on the coreset, and the result is approximately the same as on the full data set [cf. 42]. However, for some statistical problems, it is known that small coresets do not exist in the worst case. This is true, e.g., for specific generalized linear models, see the lower bounds in [37,39]. To overcome such situations, we conceptionally change the scheme. Instead of reducing the data, to approximate the full data set with respect to some model, we propose to use the statistical models derived from

✉ Leo N. Geppert
geppert@statistik.uni-dortmund.de

Katja Ickstadt
ickstadt@statistik.uni-dortmund.de

Alexander Munteanu
alexander.munteanu@tu-dortmund.de

Christian Sohler
sohler@cs.uni-koeln.de

[1] Department of Statistics, Technische Universität Dortmund, 44221 Dortmund, Germany

[2] Department of Computer Science, Technische Universität Dortmund, 44221 Dortmund, Germany

[3] Institute of Computer Science, The University of Cologne, Weyertal 121, 50931 Köln, Germany

small batches as concise summaries. Combining these statistical models via the Merge & Reduce framework, we can again turn an offline algorithm into a data stream algorithm.

We develop our work in line with the recent overview given by [51] who identified the areas of *Data Science* that lie primarily in the domains of statistics and computer science. In particular they highlight the importance of combining those areas toward meaningful statistical models with quantified uncertainty that can be obtained very efficiently from an algorithmic point of view as well as concerning data storage and access.

There are different computational models to deal with massive data sets, e.g., streaming and distributed computing. We focus on data streaming. A data stream algorithm is given an input stream of items, like numerical values, points in $\mathbb{R}^d$ or edges of a graph at a high rate. The algorithm is allowed to make only one single pass over the data. As the items arrive one by one, it maintains some sort of summary of the data that is observed so far. This can be a subsample or a summary statistic. At any time, the memory used by the algorithm is restricted to be sublinear, usually at most polylogarithmic in the number of items. For multivariate problems in $\mathbb{R}^d$ the dependence on the dimension d is often restricted to a polynomial of low degree [cf. 40].

Despite our focus on the streaming setting we stress that the Merge & Reduce scheme can also be implemented in distributed environments. Our method thus suits the criteria—identified by [52]—that need to be satisfied when dealing with *Big Data*. Specifically, the number of data items that need to be accessed at a time is only a small subset of the whole data set, particularly independent of the total number of observations. The algorithms should work on (possibly infinite) data streams. Moreover, the algorithms should be amenable to distributed computing environments like MapReduce [16].

1.1 Our contribution

As our main contribution we develop the first Merge & Reduce scheme that works directly on statistical models. We show how to design and implement this general scheme for the special cases of (Bayesian) linear models, Gaussian mixture models and generalized linear regression models. We evaluate the resulting streaming algorithms on simulated as well as real-world data sets. We hereby demonstrate that we obtain stable regression models from large data streams. Our Merge & Reduce schemes produce little overhead and are applicable in distributed settings.

1.2 Related work

Merge & Reduce was first introduced by [8] as a general method for extending static data structures to handle dynamic insertions. More recently it has been adapted to the data streaming setting, working on coresets in [2,27].

Nowadays, it is mainly employed in the design of efficient streaming and distributed algorithms for the analysis of massively large data sets. Though often implicitly mentioned, Merge & Reduce has become a standard technique in the coreset literature. Coresets have been studied extensively for nearly two decades as a data aggregation and reduction tool to address scalability issues for several problems in Computational Statistics. Coresets were developed, e.g., for shape fitting problems [1–4,23], clustering [5,21,22,35], classification [26,28,44], ℓ_2-regression [14,18,19,33], ℓ_1-regression [10,13,46], ℓ_p-regression [15,54], M-estimators [11,12] and generalized linear models [31,37,39,44,50]. See [42] for a recent and extensive survey and [38] for a technical introduction to coresets.

Merging and reducing techniques, similar to Merge & Reduce, were employed in the area of physical design for relational databases [9].

A one-pass streaming algorithm for Bayesian regression in the $n \gg d$ case was proposed by [6]. Similar to our method it reads the data blockwise and runs a number of steps via a Markov chain Monte Carlo (MCMC) sampler. When the next block is read, the algorithm keeps or replaces some of the samples. The selection criterion is based on weights that monitor the importance of the data. In our previous work [24], we developed a one-pass streaming algorithm for Bayesian regression via sketching techniques [cf. 53] based on random projections. The sketching data structure was a linear map, which makes it easy for data to be dynamically inserted or modified. New directions in (Bayesian) data analysis in the context of massive data sets are surveyed in [52].

More recently, [32] studied composable models for Bayesian analysis of streaming data. The authors address the asynchrony of sampling frequencies in practical streaming and distributed settings and thus have a different scope. However, they propose composition procedures of their Bayesian models similar to the merging procedures that we develop here in the Merge & Reduce framework.

The statistical focus in this manuscript lies on regression models. There is plenty of literature on coresets for all kinds of computational problems (see [42]) that admit Merge & Reduce schemes. This does not necessarily extend to corresponding statistical models. But since we find common design patterns like tree structures [45], hierarchical modeling [41] and challenges as accumulation and storage [47], streaming and distributed computation [49,52] in the literature, we believe that the *statistical models* tractable in Merge & Reduce can in principle be extended to classification [41,45], clustering [49] and topic modeling [7] in Big Text data mining [47]. We cannot cover this plethora of models here but point the interested researcher to the casewise design of appropriate Merge & Reduce schemes.

1.3 Preliminaries and notation

Most symbols and notations are described in the text close to their first use, but we give a centralized overview here for completeness. For any real values $a \propto b$ indicates that a is proportional to b, i.e., there exists a constant c such that $a = cb$. For any real-valued vector or matrix X, we denote its transpose by X'. Throughout the document n denotes the number of data points or observations, possibly with subscript n_b indicating the number of points in a block of data. d denotes the dimension of data points or the dimension of the regression parameter vector $\beta \in \mathbb{R}^d$, and $\hat{\beta} \in \mathbb{R}^d$ denotes the vector of estimates. In the context of regression $X \in \mathbb{R}^{n\times d}$ denotes the matrix of data points or independent variables, and $Y \in \mathbb{R}^n$ denotes the target vector of outcomes. Real-valued functions like $\exp(\cdot)$ (similarly $\ln(\cdot)$) naturally extend to vectors entrywise

$$\exp(Y) = (\exp(Y_1), \dots, \exp(Y_n))' . \tag{1}$$

Moreover, μ or $\bar{x}$ denote means, and $\tilde{x}$ denotes medians. The linear regression error is denoted ε and its standard deviation is σ_ε, while σ_j, $j \in \{1, \dots, d\}$ denote standard deviations, and s_j denote standard errors of the regression estimators. For random variables $Y \sim D$ denotes that Y follows distribution D. By $N(\mu, \sigma^2)$ we denote the (univariate) normal distribution with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \in \mathbb{R}^{>0}$ and by $N(\nu, \Sigma)$ the multivariate normal distribution with mean $\nu \in \mathbb{R}^d$ and positive definite covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. Finally e_m^2 denotes the squared Euclidean error of the regression coefficients, see Eq. 21, and f_{se}^m denotes a factor for correcting standard errors, see Eq. 22.

2 Merge & Reduce

2.1 The principle

The Merge & Reduce principle is a versatile classic technique in theoretical computer science. It allows us to perform computations on blocks of tractable size one after another and combine their results in a structured and efficient way, whenever the input data are too large to be processed as a whole. One usually starts with a sufficiently small block size n_b that fits into the main memory of the machine. The blow-up in the total memory requirements remains bounded by a factor of $O(\log n)$.

On this basis, we develop Merge & Reduce for statistical data analysis. We iteratively load as many observations into the memory as we can afford. On each of these blocks, we apply a classical algorithm to obtain, for instance, the parameters of a statistical model, some (sufficient) statistics or a summary of the presented data; in short, a *model*. Models are merged according to certain rules, eventually resulting in a final model that combines the information from all subsets. Merge & Reduce leads to stable results where every observation enters the final model with equal weight, thus ensuring that the order of the data blocks does not bias the outcome toward single observations.

In order to design a streaming algorithm for a specific statistical analysis task, we need to choose an appropriate model as a summary statistic for each block of data. The two main ingredients that we need to implement for this particular choice of a model are called *merge* and *reduce*.

1. Let M_1, M_2 be the models obtained from the analysis of data blocks B_1, B_2, then the output of **merge**(M_1, M_2) is a model M for the union $B_1 \cup B_2$ of the input data blocks.
2. Let M be a model for data block B that has become too large $|M| \geq 2T$ for some threshold T (e.g., by repeated merge operations), then **reduce**(M) computes a model M' of size $|M'| \leq T$ for the same block B.

It is not immediately clear and certainly not a trivial question how to summarize the statistical analysis and how to implement the merge and reduce functions on statistical models. The answer heavily depends on the statistical method employed and on the representation chosen to store the model. To the best of our knowledge, Merge & Reduce has not been applied directly to statistical models yet. Here, we present this novel, general concept and demonstrate it on regression models. We will discuss how to design the merge and reduce functions for linear regression in the frequentist as well as in the Bayesian setting. We will also deal with generalized linear models [36], in particular with an application to Poisson regression in Sect. 2.2. Before we get to these details we first describe how the merge and reduce functions interact in a structured way to perform the statistical analysis task on the data block-by-block while maintaining a model for the whole subset of data presented so far.

Algorithm 1 shows pseudo-code of the algorithmic scheme behind the Merge & Reduce principle. The data structure consists of $L = O(\log n/n_b) = O(\log n)$ buckets that store one statistical model each. Initially they are all empty. One bucket, the *working bucket* B_0 is dedicated to store the model for the current bunch of data, while each of the other buckets B_i stores one model on its corresponding level $i \in \{1, \dots, L\}$ of a binary tree structure formed by the merge operations (see Fig. 1). The data structure works in the following way.

First we read one block of data of size n_b. We perform the statistical data analysis on this block only. The model that summarizes the analysis is stored into B_0. We begin to propagate the model in the tree structure from bottom to top by repeatedly executing merge and reduce operations on each level. If B_1 is empty, then we just copy the model from B_0 to B_1 and empty B_0. Otherwise we have two models that

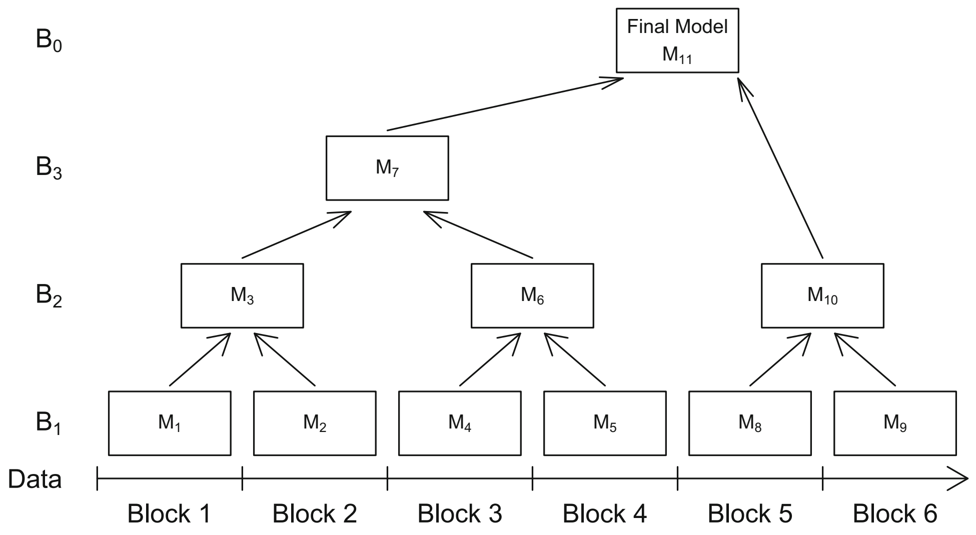

Fig. 1 Illustration of the Merge & Reduce principle. The data are presented in form of a stream and subdivided into Blocks 1 through 6 of equal size. Models M_1 to M_{11} are numbered in order of their creation throughout the execution. Arrows between models indicate the merge and reduce operations. Sibling models are deleted right after their parents' creation. Thus only one model is stored on each level, i.e., in buckets B_1 to B_3, at a time. The working bucket B_0 acts on all levels, eventually holding the final model after postprocessing at the end of the stream

Algorithm 1: Maintains a statistical model of the input data stream.

```
Input: A stream $P = (p_1, p_2, \dots p_n)$ of data points, $p_i \in \mathbb{R}^d$
Output: A summary statistic is maintained for the data stream
// initially, all $L = O(\log n)$ buckets are
   empty
for $l \in \{0, \dots, L\}$ do
    $B_l \leftarrow \emptyset$
// main loop iterates through blocks of
   size $n_b$ from the input stream
repeat
    // process one block of data
    $Q \leftarrow$ read next $n_b$ observations, less if stream ends
    $B_0 \leftarrow$ model($Q$)
    // propagate through levels of M&R-tree
    $l \leftarrow 1$
    while $B_l \neq \emptyset$ do
        $B_0 \leftarrow$ reduce(merge($B_0, B_l$))
        $B_l \leftarrow \emptyset$
        $l \leftarrow l + 1$
    // store model in lowest empty bucket
    $B_l \leftarrow B_0$
    $B_0 \leftarrow \emptyset$
until $|Q| < n_b$
// postprocessing
for $l \in \{1, \dots, L\}$ do
    $B_0 \leftarrow$ reduce(merge($B_0, B_l$))
    delete $B_l$
return $B_0$
```

are siblings in the tree, so we merge the two into B_0 and empty B_1 and proceed with B_2. Again, if it is empty, then the model from B_0 is stored in B_2 and the propagation terminates. Otherwise we have two siblings that can be merged and propagated to the next higher level in the tree. In general, the propagation stops as soon as the bucket on the current level is empty. When this happens, the update of the data structure has completed and we can move on to reading and analyzing the next block of input data. This is repeated until the end of the stream. Notice that except for the additional working bucket, we need to store at most one bucket on each level at a time since two siblings are merged immediately.

The attentive reader might ask how the reduce function comes into play. Again, that depends on the type of statistical model. For instance if the statistic is just the mean of the data and a merge operation calculates the combined mean as the (weighted) sum of the two individual means, then there is nothing to reduce, i.e., **reduce** is just the Id function, since the result has the same space complexity as each of the arguments. The situation would be different if, as an example, the summary comprises a subsample of the data of constant size C. Implementing the merge function simply as the union of the subsamples would result in sets of size $2C$ on level 2, $4C$ on level 3, generalizing to $2^{i-1}C$ on level i. On the highest level that would result in a total space complexity of $\Theta(nC)$, which is intractable. Now we might introduce the reduce operation which takes a subsample of size C of its argument to ensure that complexities of the models do not increase. The effect is that after each merge operation that doubles the size of a summary to $2C$, the reduce operation reduces the model again to size C. The total space complexity of the Merge & Reduce structure remains bounded by $O(C \log n)$.

It is also important to note that every time two models are merged and reduced, we can incur an error that can accumulate on the path from a leaf of raw data to the root comprising the final model. By merging only siblings in the streaming phase we establish a binary tree structure with bounded height $h = O(\log n)$. This ensures that in their trade-off, memory overhead and cumulated errors *both* remain bounded by at most logarithmic factors.

When the stream of input data has come to an end, we need to go through the whole list of buckets once more and merge (and reduce, if necessary) them one-by-one into one model for the whole data set. We will refer to this as the *postprocessing*. In the postprocessing phase the binary tree structure is already established and its height is bounded such that merging across all $O(\log n)$ levels can be applied arbitrarily.

Before running the Merge & Reduce scheme, it is necessary to set the number of observations per block to a number n_b. In a streaming or Big Data setting, the total number of observations n is typically not known beforehand. For that reason, n_b does not depend on n, but mainly on memory limitations of the computing device or space requirements of performing the analysis and similar considerations. The total number of blocks and levels L, respectively, is then also unknown beforehand. We thus start with only two buckets B_0, B_1 and extend the list of buckets for the higher levels as it becomes necessary.

In our implementations, every model summarizes the respective regression analysis of a block or the union of neighboring blocks. Additionally, all models contain meta-information; specifically the number of observations the model is based on. This number is n_b for the models on level 1, except for the last block of data which may be smaller. Whenever two models are merged, the resulting number is just the sum of observations that the two siblings are based upon. Merging two models that are not based on the same number of observations may occur when the last block of input data is involved or in the postprocessing phase. This does not pose a problem provided appropriate weighting is employed in the merge operation. We will also address this issue in Sect. 2.2.

Example of Merge & Reduce execution Note that the propagation of models through the $O(\log n)$ levels mimics the behavior of a binary counter, as given in pseudocode in Algorithm 1. However, it might be more accessible to think of the calculations being done in a postorder traversal of a binary tree structure. Figure 1 illustrates the principle of Merge & Reduce from this perspective. Here, all models are numbered in the order in which they are generated in a sequential data streaming application. First, Block 1 comprising n_b observations is read from the stream into the memory and the model M_1 is derived from its statistical analysis. The same process yields model M_2 derived from the data contained in Block 2 of the stream. Since M_1, M_2 are siblings in the tree, they are combined into $M_3 := \textbf{reduce}(\textbf{merge}(M_1, M_2))$. At this point M_1, M_2 are not necessary anymore and deleted from memory. The merge and reduce operations are indicated by the arrows in Fig. 1. Now we proceed with M_4 derived from Block 3 and M_5 derived from Block 4. Since M_4, M_5 are siblings in the tree, they are combined into $M_6 := \textbf{reduce}(\textbf{merge}(M_4, M_5))$ and deleted. Again we have siblings M_3, M_6 on the same level which are combined to $M_7 := \textbf{reduce}(\textbf{merge}(M_3, M_6))$ and deleted thereafter. The procedure is continued in the same manner until the stream has reached its end. Say this is the case after processing Block 6. Note that at this point M_8, M_9 have been merged and reduced into M_{10} and have been deleted. The current state of the data structure is that it holds only Models M_{10} in bucket B_2, i.e., on level 2, and M_7 in bucket B_3, i.e., on level 3, respectively. The buckets B_0 and B_1 are empty at this point, and there are no further levels above level 3. Now the postprocessing step implicitly merges $M_{11} = \textbf{reduce}(\textbf{merge}(M_7, M_{10}))$ via the working bucket B_0.

The construction can also be computed in a parallel or distributed setting. One possible scheme to achieve this is to compute all models on the same level in parallel, starting with models M_1, M_2 M_4, M_5, M_8, M_9 on level 1 and proceeding with parallel computation of M_3, M_6, M_{10} on level 2 followed by M_7 on level 3 and finally deriving the final model M_{11} from M_7, M_{10}. Our work thus meets the criteria that were proposed by [52] for the large n case in streaming as well as distributed computational environments. We focus on the sequential streaming setting in the remainder for brevity of presentation.

2.2 Combination of Merge & Reduce with regression analysis

We will now give a short introduction to different regression methods that we are going to cover in this article. Thereafter we are going to derive three different Merge & Reduce schemes to implement the algorithmic approach that we have introduced in previous Sect. 2.1.

Following [25], a linear regression model is given by

$$Y = X\beta + \varepsilon, \tag{2}$$

where $Y \in \mathbb{R}^n$ is the dependent variable and $X \in \mathbb{R}^{n\times d}$ is the design matrix, which contains the observations of the independent variables $x_1, \ldots, x_d$. The first column of the design matrix often consists of 1-entries to model an intercept. X may also contain transformations of or interactions between variables. The error term ε is assumed to be unobservable and nonsystematic, usually with the additional assumption of a normal distribution $N(0, \sigma_\varepsilon^2)$. The last component, $\beta \in \mathbb{R}^d$, is an unknown vector whose true value we wish to estimate.

In a frequentist setting, β is a fixed but unknown quantity. Equation 2 can be solved for β using the ordinary least squares estimator

$$\hat{\beta} = (X'X)^{-1}X'Y, \tag{3}$$

in the sense that it minimizes the sum of squared residual errors, i.e., $\hat{\beta} = \operatorname{argmin}_{\beta\in\mathbb{R}} \|X\beta - Y\|_2^2$.

In a Bayesian setting, β is assumed to be a random vector and thus to follow a distribution itself. In addition to the assumption of a Gaussian likelihood, incorporating a priori knowledge into the model is possible by defining a prior distribution of β, $p_{\text{pre}}(\beta)$. The likelihood models the information given in the data, while the prior models knowledge that may not be present in X or Y. We are interested in the posterior distribution

$$p_{\text{post}}(\beta|X, Y) \propto \mathcal{L}(\beta|X, Y) \cdot p_{\text{pre}}(\beta). \tag{4}$$

From Eq. 4, we can immediately see that the posterior distribution $p_{\text{post}}(\beta|X, Y)$ is a compromise between the likelihood $\mathcal{L}(\beta|X, Y)$, given the observed data, and the prior distribution $p_{\text{pre}}(\beta)$. For a small class of models, it is possible to calculate the posterior distribution analytically. In most cases, though, it is necessary to employ approximation techniques. We will utilize Markov chain Monte Carlo (MCMC) methods in this manuscript, more specifically a modern MCMC algorithm called Hamiltonian Monte Carlo [30].

Generalized linear models (GLMs) are a class of models extending strictly linear models [36]. They offer more flexibility by employing a nonlinear link function g, which serves as connection between the expected value $E(Y)$ of Y and the linear predictor $X\beta$. This results in

$$g(E(Y)) = X\beta. \tag{5}$$

Common cases of GLMs are logistic regression, where Y is a binary variable, and Poisson regression, where Y is a count variable. Linear regression can also be seen as a special case of a GLM, where the link function g is the identity function. In the present paper, we concentrate on Poisson regression. While other link functions can be chosen, the so-called canonical link function is the natural logarithm, resulting in $\ln(E(Y)) = X\beta$ or, equivalently, $E(Y) = \exp(X\beta)$. Poisson regression can be carried out both in a frequentist and a Bayesian setting.

2.2.1 Implementation in Merge & Reduce

Here we develop concrete approaches of our conceptually novel Merge & Reduce scheme that acts directly on statistical models. We will now define our statistical summaries and the required operations for conducting the analyses within the Merge & Reduce framework. When looking at the output that results from the different statistical analyses, the main difference is whether it is a frequentist or a Bayesian model. A special case that allows analytical calculations is linear regression with Gaussian errors. For this reason, we develop three Merge & Reduce approaches.

Merge & Reduce approach 1 (general frequentist models): The standard procedure for frequentist linear regression as well as for GLMs return an estimate $\hat{\beta}$, an estimated standard error $\hat{s}_j$ for every component $\hat{\beta}_j$, $j = 1, \ldots, d$ as well as a test statistic and a p-value derived from the estimate and its standard error. To summarize these values, it is sufficient to include the estimate and the standard error in the model. Our summary statistic is thus a vector

$$S = (\hat{\beta}_1, \ldots, \hat{\beta}_d, \hat{s}_1, \ldots, \hat{s}_d), \tag{6}$$

together with the number of observations n_i it is based on. Two blocks are merged by calculating the weighted means of their respective vectors S (see below for a formal introduction). Merge & Reduce (M&R) approach 1 thus leads to an unbiased estimation of β provided $\hat{\beta}$ is an unbiased estimate of β in every block. This is the case provided the regression model employed is suitable for all observations and an appropriate estimate is available, which is the case for a wide variety of regression models.

The estimated standard error $\hat{s}$ usually depends on the data, e.g., for a linear regression model,

$$\hat{s}_j = \sqrt{\sigma_\varepsilon^2 e_j'(X'X)^{-1} e_j}, \quad j = 1, \ldots, d, \tag{7}$$

where $e_j \in \mathbb{R}^d$ is the j^{th} unit vector. Assuming that the values of X are realizations from a random distribution with expected value $-\infty < \mu_X < \infty$ and variance $\sigma_X^2 < \infty$ for all blocks, the standard errors in the reduced data sets are inflated compared to those on the full data set by a factor of $\sqrt{\lceil n/n_b \rceil}$. This can easily be corrected for. If there are systematic differences between the values of X for different blocks, the standard error will be underestimated even after correction.

Merge & Reduce approach 2 (general Bayesian models): In the Bayesian case, the result is a distribution, which is approximated by an MCMC sample. The summary values should be chosen such that they characterize the distribution in a concise way. In this case, more than one reasonable and useful solution is possible. In the present manuscript, we utilize the mean $\bar{x}_j$ and median $\tilde{x}_{.5}$, the upper and lower quartiles $\tilde{x}_{.25}$, $\tilde{x}_{.75}$, the 2.5% and 97.5% quantiles $\tilde{x}_{.025}$, $\tilde{x}_{.975}$ and the standard deviations σ_j of the MCMC sample for every component β_j as summary values. This choice gives a good indication of the location and variation of the posterior distribution. However, for some purposes, other summary values may offer a better representation of the results. Careful consideration of the requirements of one's analysis is recommended. Our summary statistic again is a vector

$$S = (\bar{x}_1, \ldots, \bar{x}_d, \tilde{x}_{p,1}, \ldots, \tilde{x}_{p,d}, \sigma_1, \ldots, \sigma_d), \tag{8}$$

where $p \in \{.025, .25, .5, .75, .975\}$ iterates through the quantiles detailed above. We also store the number of observations n_i which S is based on.

Merge operation (common to Merge & Reduce approaches 1,2): For both M&R approaches 1 and 2, we propose conducting the merge step by taking the weighted mean for every summary value considered. To this end, let S_1 and S_2 be the vectors of summary values for models that we want to merge. The weights are chosen proportional to the number of observations n_1 and n_2 the models are based on, respectively. The new, merged vector of summary values S_m and observation number n_m is then obtained by

$$S_m = w_1 S_1 + w_2 S_2, \tag{9}$$

$$n_m = n_1 + n_2, \tag{10}$$

where the weights w_1 and w_2 are given by $w_1 = \frac{n_1}{n_m}$ and $w_2 = \frac{n_2}{n_m}$.

Merge & Reduce approach 3 (linear regression with Gaussian error): In the case of frequentist linear regression where we can assume normality for the distribution of the estimators, building products of distributions offers another way of merging and reducing the models. Let f_1, f_2 be the density functions of $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$, respectively. Their pointwise product $f_P(x) = f_1(x) \cdot f_2(x)$ is proportional to a normal distribution $N(\mu_P, \Sigma_P)$ where the parameters are obtained by

$$\Sigma_P^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \tag{11}$$

$$\Sigma_P^{-1}\mu_P = \Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2. \tag{12}$$

The parameters are implicitly weighted via the inverse covariance matrices.

Instead of directly calculating the means and covariance matrices, we propose maintaining $X_i'X_i$, $X_i'Y_i$, $Y_i'Y_i$ and n_i as summary statistics for each block i. These quantities enable us to recover the ordinary least squares estimator (Eq. 3) and its inverse covariance matrix $\Sigma^{-1} = \frac{1}{\sigma_\varepsilon^2} X_i'X_i$ [cf. 25]. Note that the factor σ_ε^2 is not explicitly stored until the end of computations. It can be estimated blockwise as well, but since it is present in each term of Eq. 12, it can be factored out and ignored until the last step; see below for details.

Merge operation (Merge & Reduce approach 3):

For M&R approach 3, we propose the following merge operation. Given data $[X_m, Y_m]$, each divided into two blocks, $X_m = [X_1', X_2']'$ and $Y_m = [Y_1', Y_2']'$, we can merge their summaries via

$$X_m'X_m = X_1'X_1 + X_2'X_2, \tag{13}$$

$$Y_m'Y_m = Y_1'Y_1 + Y_2'Y_2, \tag{14}$$

$$Y_m'X_m = Y_1'X_1 + Y_2'X_2, \tag{15}$$

$$n_m = n_1 + n_2. \tag{16}$$

From this, we can reconstruct $N(\mu_P, \Sigma_P)$ for the combined block. The maximum likelihood estimate for the mean [cf. 25] is obtained via the ordinary least squares estimator

$$\mu_P = \hat{\beta}_m = (X_m'X_m)^{-1}X_m'Y_m$$
$$= (X_m'X_m)^{-1}(Y_m'X_m)' \tag{17}$$

(cf. Eq. 3). The maximum likelihood estimate of the model variance [cf. 25] is given by

$$\hat{\sigma}_\varepsilon^2 = \frac{\|X_m\hat{\beta} - Y_m\|_2^2}{n_m - d}, \tag{18}$$

where

$$\|X_m\hat{\beta} - Y_m\|_2^2$$
$$= \hat{\beta}_m'X_m'X_m\hat{\beta}_m + Y_m'Y_m - 2Y_m'X_m\hat{\beta}_m. \tag{19}$$

Hence, the maximum likelihood estimate of the covariance matrix can be computed from the summary as

$$\Sigma_P = \hat{\sigma}_\varepsilon^2 (X_m'X_m)^{-1}. \tag{20}$$

Reduce operation (common to all): The above merge operations ensure by construction that the memory requirement of merged models does not increase. The reduce step is thus inherently included in the merge step and can be interpreted as Id function as discussed previously.

3 Simulation study

Here we assess the performance of the newly developed methods empirically. To that end, we conduct a series of simulations with the aim of comparing the results of the original regression model and the summary values obtained employing Merge & Reduce for both frequentist and Bayesian regression. The data sets were created and analyzed using R, versions 3.1.2, 3.4.1 and 3.4.4 [43]. In addition to that, the R-package `rstan`, version 2.14.1 [48], was employed for the analysis of all Bayesian regression models. We are developing an R package that provides functionality to apply Merge & Reduce on regression models. This package `mrregression` will be available on CRAN.

3.1 Data generation

The main parameters used to generate data sets for the simulation study are the number of observations n, the number of variables d and the standard deviation of the error term σ_ε. Different numbers of observations per block n_b are also chosen. The size of n_b does not influence the data generation, but may have an influence on the outcome. An overview of the range of values is given in Table 1. The setup of the simulation study and the parameter values are chosen similar to the simulation study in [24]. Simulated data sets where the number of observations is less than the number of observations per block, $n < n_b$, are excluded from the simulation study. Please note that we include very small block sizes of $n_b = 400$ and $n_b = 1000$ to examine the limitations of the method that will be discussed in detail later on.

In addition to these parameters, we consider three different scenarios. In the first scenario, all assumptions of a linear regression model are met. A varying fraction of the variables has an influence (large or small) on the dependent variable while the remainder is not important for the explanation of Y. The aim in this scenario is to obtain a general overview of the roles the parameters play, especially the effect of n_b. This situation is covered in Sect. 3.2.

In the second scenario, each data set consists of two mixtures, i.e. it has two components where n_1 simulated observations stem from the first mixture component and n_2 observations from the other ($n_1 \geq n_2, n_1 + n_2 = n$). In our simulation study, we choose $\frac{n_2}{n} = 0.05$. For that reason, observations belonging to the second mixture component can be considered as outliers. These outliers may differ with regard to X as well as Y. However, both the main body of observations and the outliers follow the same regression model. For the second scenario, the two components are put together in four different ways to simulate different orders—invariable to the data analyst—in which the data are presented to the Merge & Reduce algorithm:

`first`	outliers first, followed by main body of observations.
`last`	main body first, followed by outliers;
`middle`	first half of observations from main body, followed by outliers, followed by second half of observations from main body;
`random`	random arrangement;

We pursue two aims with the analysis of this scenario. As mentioned in Sect. 2.1, the order of blocks should not influence the outcome of the Merge & Reduce result. In our simulation study, we do not change the order of the blocks, but the order of the observations. Some differences can therefore be expected, but the results should be comparable for the first three orders. Additionally, we include both models with intercept and without an intercept to investigate the difference between linear and affine functions.

This setup also constitutes a mild adversarial example. It is mild, because the underlying true β remains the same for the whole data set, but it is adversarial, because we both X and Y now stem from two different data distributions, leading to additional variation in the data. If the outlying observations are ordered randomly, both data distributions are expected to be present in each block. For the other three orders, one or two structural breaks are present in the data set with regard to X and Y. We expect these cases to have an inflating effect on the estimated standard errors for Merge & Reduce, because the variation in most blocks is less than on the whole data set by construction. The results from the second scenario are discussed in Sect. 3.3.

In the third scenario, we generate data sets with count data as dependent variable, i.e., $Y_i \in \mathbb{N}_0, i = 1, \ldots, n$. The true β is simulated as before. For each observation, the expected value is computed according to $\exp(X\beta)$. Finally, the values of y_i are obtained by drawing n random numbers from Poisson distributions with the respective expected values. On these data sets, Poisson regression is employed and all the assumptions are met. The parameter σ_ε is not applicable in this scenario as the variance is equal to the expected value by definition. Section 3.4 presents the outcomes found in this scenario.

3.2 Linear regression

In this section, we analyze simulated data sets where all assumptions of a linear model are fulfilled. The aim is to gain insight into the roles of the parameters n, d, σ_ε, and n_b (see Table 1). In the frequentist case, linear models with Y as dependent variable and $X_1, \ldots, X_d$ as independent variables (possibly including an intercept term) are calculated for the whole data sets using function `lm`. The same data sets are also analyzed using M&R approaches 1 and 3 (confer Sect. 2.2), again employing `lm` to obtain the linear model in each block. In the Bayesian case, we employ a standard linear model with parameters $\beta_1, \ldots, \beta_d$ and σ_ε.

M&R approach 1

In the following, we will evaluate M&R approach 1 using two criteria for all models m in our simulation study ($m = 1, \ldots, M$). The first criterion e_m^2 is the squared Euclidean distance between the Gauß–Markov estimate $\hat{\beta}^m_{orig}$ and the vector $\hat{\beta}^m_{MR}$, obtained from the Merge & Reduce algorithm,

$$e_m^2 = \|\hat{\beta}^m_{MR} - \hat{\beta}^m_{orig}\|_2^2, m = 1, \ldots, M. \tag{21}$$

Values of e_m^2 close to zero are desirable as this means Merge and Reduce is able to recover the original model closely. The second criterion is the corrected standard error factor

$$f_{se}^m = \left(\frac{1}{d}\sum_{j=1}^{d}\frac{s^m_{j,MR}}{s^m_{j,orig}}\right) \Big/ \sqrt{\left\lceil\frac{n}{n_b}\right\rceil}, \tag{22}$$

Table 1 Main parameters in the simulation study

Parameter	Role	Range
n	Number of observations	20,000–1,000,000
d	Number of variables	5–200
σ_ε	Standard deviation of error term	0.1–10
n_b	Observations per block	400–25,000

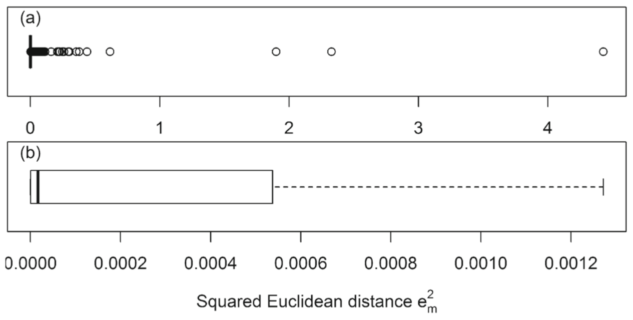

Fig. 2 Boxplots of squared Euclidean distances between M&R approach 1 and original model observed in the simulation study, across all values of n, d, σ_ε, and n_b. **a** Contains all values, **b** excludes 180 out of 952 values that lie further than 1.5 times the interquartile range away from the upper quantile

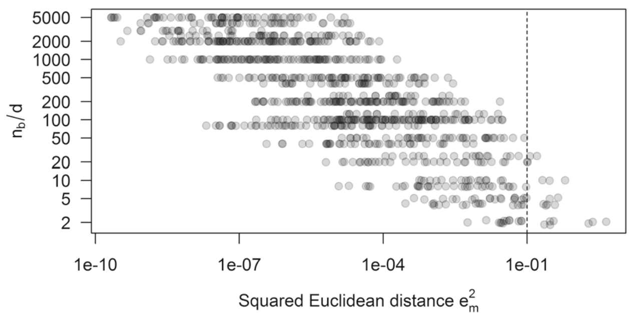

Fig. 3 Scatterplot of the effect of observations per block per variable $\frac{n_b}{d}$ on squared Euclidean distances e_m^2, $(m = 1, \ldots, M)$ for M&R approach 1. x- and y-axes are on a logarithmic scale, and observations are drawn as partially transparent points: gray points mean single observations, black points multiple observations at roughly the same location. Vertical dashed line is at 0.1

where $s^m_{j,orig}$ and $s^m_{j,MR}$ are the estimated standard errors for variable j, $(j = 1, \ldots, d)$ according to the maximum likelihood estimate and M&R approach 1. Here, a value of 1 indicates that the estimated standard errors are the same for all variables, while values $f^m_{se} > 1$ indicate an inflated estimated standard error for M&R approach 1. Using the mean over all $\frac{s^m_{j,MR}}{s^m_{j,orig}}$ in Eq. 22 may seem surprising. In our simulation study, these ratios are almost identical for all variables of a model m so that using the mean or median or any other measure of location would return similar results.

Figure 2 shows boxplots for all values of e_m^2 we observed in the simulation study. Subfigure (a) contains every value and is visually dominated by the outliers. Only 3 out of 952 values exhibit noticeably high levels of e_m^2 with values between about 2 and 4.5. The majority of values of e_m^2 seems to be close to 0 with some outliers between 0 and 1. Subfigure (b) shows only the box of the same boxplot, with all outliers removed. Here, it becomes clear that 75% of the observed values of e_m^2 lie between 0 and 0.00054. In total, 180 values above 0.00127 are considered outliers as they are farther than 1.5 times the interquartile range away from the upper quartile.

For the most part, M&R approach 1 is able to recover the original linear models well. However, there are cases with unacceptably high values of e_m^2 which will lead to deviations from the original model. Further analysis of the results indicates that this is driven by two mechanisms. First, the ratio between the number of observations per block n_b and the number of variables d plays a critical role. In our simulation study, for some data sets this ratio is only two. As Fig. 3 depicts, for a majority of the cases, this leads to values of $e_m^2 < 0.1$, but all cases of exceptionally high values of e_m^2 are found when the ratio of n_b and d is 2. As $\frac{n_b}{d}$ increases, the portion of high values of e_m^2 tends to decrease, and when $\frac{n_b}{d} > 25$, all values of e_m^2 are less than 0.1.

Second, e_m^2 is dominated by one component of the squared Euclidean distance, the intercept. Especially for the simulated data sets which lead to high e_m^2 values, the intercept accounts for at least 70% of the distance in all cases and for 90% in most cases.

The corrected standard error factor f^m_{se} also shows a dependency on $\frac{n_b}{d}$. As Fig. 4a shows, a great majority of the values of f^m_{se} is close to 1, which ensures that the original model and the Merge & Reduce model return similar results. There are, however, some data sets where the estimated standard errors for the Merge & Reduce result are inflated by more than 40% compared to the original model.

Figure 4b breaks the corrected standard error factor f^m_{se} down according to the ratio of observations per block and variables in the data set. There is a clear connection between $\frac{n_b}{d}$ and f^m_{se}. For low values of $\frac{n_b}{d}$, we can observe the highest values of f^m_{se}. As $\frac{n_b}{d}$ increases, the corrected standard error factor decreases. If every block contains at least 25 observations per variable, the estimated standard errors were inflated by less than 2.5% in our simulation study.

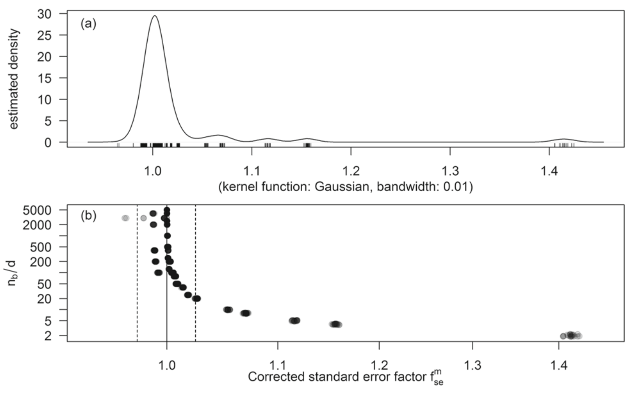

Fig. 4 **a** A kernel density estimate of corrected standard error factors f^m_{se} for M&R approach 1 across all settings. **b** A scatterplot of the effect of observations per block per variable $\frac{n_b}{d}$ on corrected standard error factors f^m_{se} across all other settings for M&R approach 1. y-axis is on a logarithmic scale, and observations are drawn as partially transparent points: gray points mean single observations, black points multiple observations at roughly the same location. The solid vertical line indicates values of $f^m_{se} = 1$, and the two dotted lines stand for relative deviations of 2.5%, i.e. $f^m_{se} = 0.975$ and $f^m_{se} = 1.025$

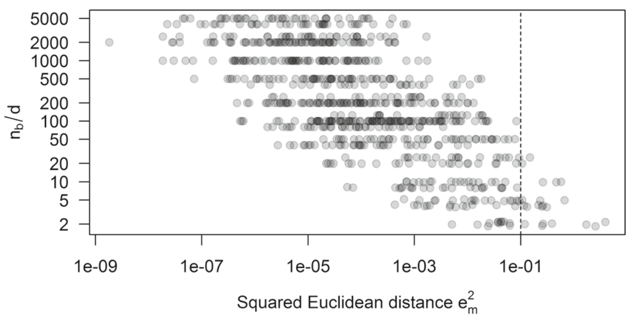

Fig. 5 Scatterplot of the effect of observations per block per variable $\frac{n_b}{d}$ on squared Euclidean distances e^2_m, $(m = 1, \ldots, M)$ between posterior medians for M&R approach 2. x- and y-axes are drawn on a logarithmic scale, and observations are drawn as partially transparent points: gray points mean single observations, black points multiple observations at roughly the same location. Vertical dashed line is at 0.1

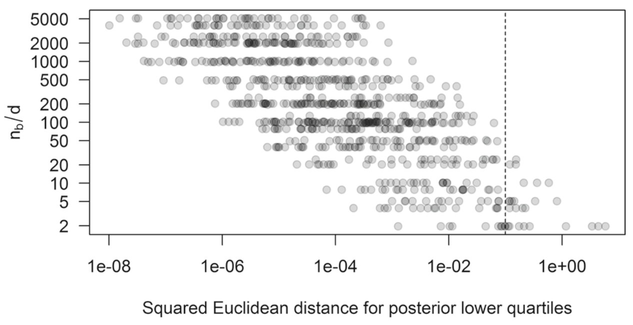

Fig. 6 Scatterplot of the effect of observations per block per variable $\frac{n_b}{d}$ on squared Euclidean distances between posterior lower quartiles for M&R approach 2. Both x- and y-axis are on a logarithmic scale, and observations are drawn as partially transparent points: gray points mean single observations, and black points multiple observations at roughly the same location. Vertical dashed line is at 0.1

Some simulated data sets exhibit values of f^m_{se} less than 1. These are cases, where $\frac{n}{n_b}$ does not result in an integer number. In Eq. 22, rounding up of that fraction occurs, which seems to result in an overcorrection. However, this leads to an underestimation of the estimated standard error of at most 2%, which still approximates the result of the original linear model well. There are two exceptions with an underestimation of around 3.5%, visible in the upper left corner. These are two simulated data sets with $n = 20{,}000$ and $d = 5$ where the number of observations per block $n_b = 15{,}000$ is very close to n.

In conclusion, M&R approach 1 recovers both estimates and standard errors of the original linear model well, provided, the ratio of observations per block and variables, $\frac{n_b}{d}$, is large enough. Based on the results of our simulation study, we recommend $\frac{n_b}{d} > 25$. [29] recommends at least 10 or 20 observations per variable for linear models to be considered reliable. Merge & Reduce seems to increase the required number of observations per variable, but in a very moderate way. In addition, Eq. 22 seems to lead to a slight overcorrection and thus underestimation of the standard error when the number of observations per block n_b is close to the total number of observations n in the data set. In a Big Data setting, both constraints do not pose a problem.

M&R approach 2

In the Bayesian case, we consider both measures of location and measures of dispersion. Here, we evaluate all of the summary values by calculating the squared Euclidean distance to the corresponding summary value resulting from the original model. For measures of location, we employ the values from the MCMC sample representing the posterior distribution directly, similar to Eq. 21. For quantiles that represent measures of dispersion such as quartiles and $p_{0.025}$ and $p_{0.975}$, we introduce a correction,

$$\tilde{x}^{\text{corr}}_{p,j} = \frac{\tilde{x}_{p,j} - \bar{x}_j}{\sqrt{\lceil n/n_b \rceil}} + \bar{x}_j. \tag{23}$$

The only exception to this is the posterior standard deviation, which we correct using the factor $\sqrt{\lceil n/n_b \rceil}$ as in Eq. 22.

Figure 5 shows a plot containing the squared Euclidean distances between the posterior median based on M&R approach 2 and the original Bayesian model for all simulated data sets grouped according to the ratio $\frac{n_b}{d}$. Similarly to M&R approach 1, for a small number of simulation settings we observe values of e^2_m of up to almost 4, but the majority of squared distances is close to 0. The distances exhibit the same pattern we observed in the frequentist case, i.e., the median is well recovered by M&R approach 2 provided $\frac{n_b}{d} > 25$. Results for the posterior mean are almost identical.

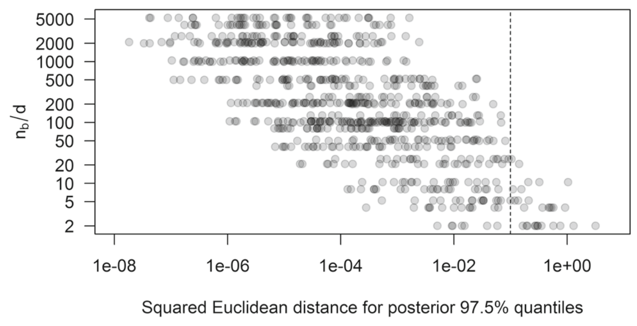

Fig. 7 Scatterplot of the effect of observations per block per variable $\frac{n_b}{d}$ on squared Euclidean distances between posterior 97.5% quantiles for M&R approach 2. Both x- and y-axis are on a logarithmic scale, and observations are drawn as partially transparent points: gray points mean single observations, and black points multiple observations at roughly the same location. Vertical dashed line is at 0.1

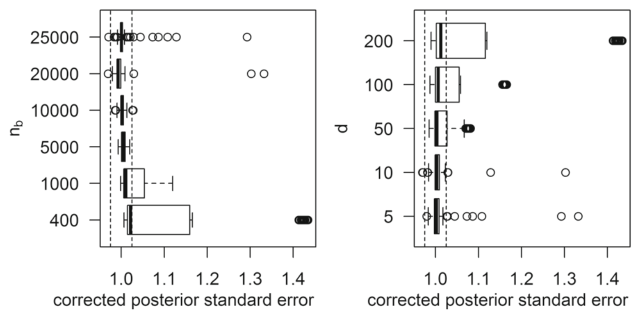

Fig. 8 Boxplots of corrected standard error factor f_{se}^m based on the posterior standard deviation. **a** Effect of block size n_b on f_{se}^m, **b** effect of the number of variables d on f_{se}^m. The two dotted lines stand for relative deviations of 2.5%, i.e., $f_{se}^m = 0.975$ and $f_{se}^m = 1.025$

For $\tilde{x}_{0.025}$, $\tilde{x}_{0.25}$, $\tilde{x}_{0.75}$, and $\tilde{x}_{0.975}$, we employ the correction given in Eq. 23. This leads to a good recovery of the quantiles provided the ratio of observations per block and variables is high enough. Figures 6 and 7 show the squared Euclidean distances between the original model and the result of M&R approach 2 for the posterior lower quartiles and 97.5% quantiles in conjunction with the number of observations per block per variable. Similar to the results observed before, the distances grow as the ratio $\frac{n_b}{d}$ grows. For values of $\frac{n_b}{d} > 25$, in our simulation study all observed distances are lower than 0.1, leading to results that are very close to the original model.

The posterior standard deviation seems to depend on the value of n_b even after applying the correction factor. Figure 8 indicates that only for values of $n_b = 5000$, all values of the corrected standard deviation fall into the interval [0.975, 1.025]. Even for $n_b = 20{,}000$ and $n_b = 25{,}000$, cases of undesirably highly inflated posterior standard deviation occur. Contrary to results in the frequentist case, high values of f_{se}^m can also be found for a low number of variables d and thus for high values of observations per block per variable $\frac{n_b}{d}$. Because of this, we would recommend not employing the posterior standard deviation in a Bayesian Merge & Reduce setting. Instead, using suitable posterior quantiles is preferable.

M&R approach 3

M&R approach 3 works exceptionally well in the case where all assumptions are met. Employing the same simulated data sets as for M&R approach 1, we find that all squared Euclidean distances e_m^2 are 0 for $m = 1, \ldots, M$. Both estimators $\hat{\beta}_{MR}$ and $\hat{\beta}_{orig}$ thus exhibit the same values for all variables, up to numerical precision of the machine. Similarly, the standard error factor f_{se}^m lies in a range of [0.998, 1] for all simulated data sets. The small differences are due to employing the biased maximum likelihood estimator to estimate the variance. The results indicate that M&R approach 3 is able to recover the estimated standard error of the linear model perfectly.

3.3 Linear regression in the presence of mixtures

Following the analysis of the three different M&R approaches for standard linear models, we will now examine only M&R approaches 1 and 3 in the context of mixture distributions. The simulated data sets are made up of two components, where the second component, the *outliers*, may show unusual values in all of the variables. Still, the linear model used to obtain the y-values is valid for both components.

The order of items in the simulated data set is varied to simulate differently ordered data streams presented to the Merge & Reduce algorithm. As explained in Sect. 3.1, the order is usually invariable to the data analyst in a streaming setting. The four different orders are from now on referred to as *position*, where the values stand for the position of the outlying observations:

`first`	outliers first, followed by main body of observations.
`last`	main body first, followed by outliers;
`middle`	first half of observations from main body, followed by outliers, followed by second half of observations from main body;
`random`	random arrangement;

As in Sect. 3.2, M&R approaches 1 and 3, we will compare the values of $\hat{\beta}$ according to Merge & Reduce and the original model using the squared Euclidean distance and the estimated standard errors via the corrected standard error factor. In the current section, the ratio of observations per block and number of variables is greater than 50 for all settings.

Figure 9 shows the squared Euclidean distance between the estimated parameter vectors split up by the position of the outlying observations in the data set. The distances are generally small, but some settings lead to values above 0.1. There seems to be different behavior depending on the posi-

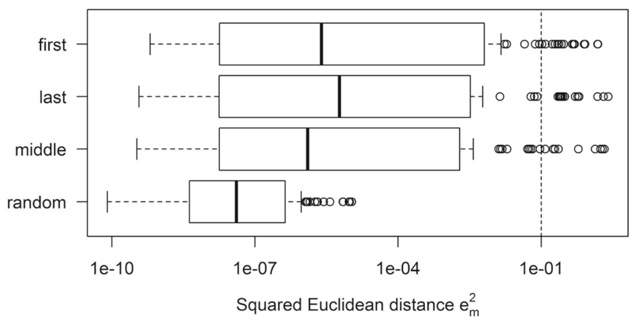

Fig. 9 Boxplots of the effect of position of the outlying observations on squared Euclidean distance e_m^2 in mixture scenario for M&R approach 1. y-axis is on a logarithmic scale, and vertical dashed line is at 0.1

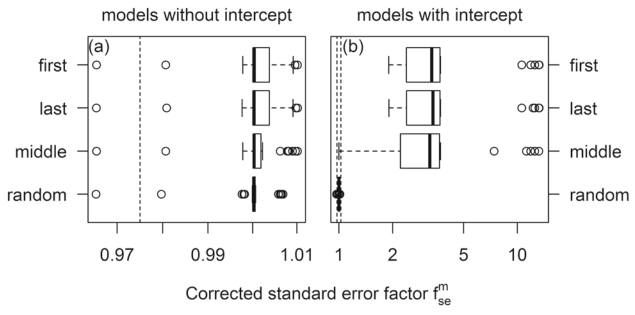

Fig. 10 Boxplots of the effect of position of the outlying observations on standard error factor f_{se}^m in mixture scenario for M&R approach 1. **a** Contains models and data sets without intercept term, **b** contains models and data sets with an intercept term. y-axis is on a logarithmic scale, and vertical dashed lines are at 0.975 and 1.025

tion of the group of outliers. Data sets with random allocation of the outlying observations seem to exhibit systematically smaller distances between the estimated parameter vectors. The other three ways of putting the data sets together exhibit larger values of e_m^2 and may also lead to squared Euclidean distances above 0.1. This indicates that Merge & Reduce is better able to recover the results of the original model in cases where the outliers are distributed across more blocks.

Figure 10 shows the corrected standard factor f_{se}^m. Here, the results depend not only on the position, but in particular on whether the model comprises an intercept term. For models without intercept, the corrected standard error factor is close to 1 for all settings of the simulation study, indicating that the estimated standard error is the same for both the original model and the model based on Merge & Reduce after correction. If the model does contain an intercept term, the corrected standard error factor is only close to 1 if the outliers are randomly inserted into the data set. If the outliers occur consecutively in the data set, the standard error is overestimated by the Merge & Reduce technique. The overestimation is considerably higher for the intercept term, but overestimation of around 8-10% also occurs for the other variables. This indicates that M&R approach 1 only works reliably for data with outliers that follow the same model if the outliers are distributed evenly around the data set or the model contains no intercept term.

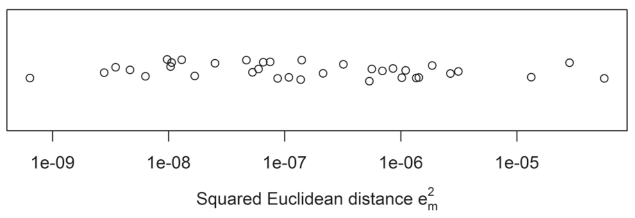

Fig. 11 Stripchart (one-dimensional scatterplot) of squared Euclidean distances e_m^2 for all Poisson regression models for M&R approach 1. x-axis is on a logarithmic scale. Vertical dashed line at 0.1 is not visible due to small values of e_m^2. Values on y-axis are jittered, i.e., small values of random noise are added

For M&R approach 3, however, the simulation study again indicates that the original models can be perfectly recovered, even when mixtures are present. Even though the presence of mixtures may increase the variance for some blocks but not for others, this is taken care of by the merge operations. When analyzing a frequentist linear regression model with Merge & Reduce, it is thus advantageous to employ M&R approach 3.

3.4 Poisson regression

To analyze the results of the Poisson regression models, we consider M&R approaches 1 and 2. The simulation study for this case contains a total of six data sets with values of $n \in \{50{,}000, 100{,}000\}$, $d \in \{5, 10, 20\}$, and $n_b \in \{400, 1000, 5000, 10{,}000, 20{,}000, 25{,}000\}$. The error term variance for Poisson-distributed data is equal to its mean and is thus not eligible.

To evaluate how well the Merge & Reduce technique is able to recover the results of the original model, we again employ squared Euclidean distance (Eq. 21) for the parameter estimates in the frequentist case as well as for all summary values in the Bayesian case. Corrected standard error factor (Eq. 22) is used for the standard error in the frequentist case and additionally for the posterior standard deviation in the Bayesian case.

M&R approach 1

Figure 11 shows the squared Euclidean distances for all Poisson models in the simulation study. We can clearly see that all distances take very low values, even for the lower values of observations per block per variable, which are $\frac{n_b}{d} = 20$ for the Poisson regression models.

Figure 12 shows the corrected standard error factor. Here, all but two values are within the acceptable range of [0.975, 1.025]. The two values outside of that range come from simulated data sets with $n_b = 400$ and $d = 20$, showing that the standard error factor exhibits a greater dependence on $\frac{n_b}{d}$ than the parameter estimate. The next observations come

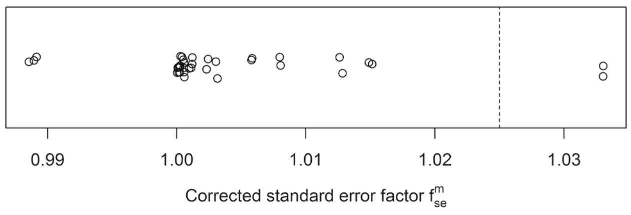

Fig. 12 Stripchart of corrected standard error factors f_{se}^m for all Poisson regression models for M&R approach 1. Vertical dashed line is at 1.025. Values on y-axis are jittered

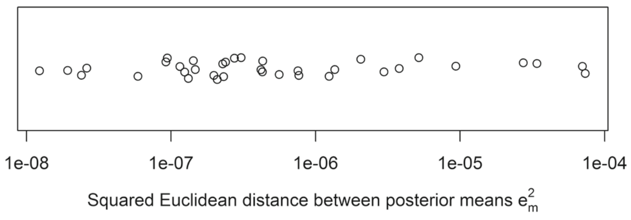

Fig. 13 Stripchart of squared Euclidean distances between posterior means of original model and Merge & Reduce model for all Bayesian Poisson regression models for M&R approach 2. x-axis is on a logarithmic scale. Vertical dashed line at 0.1 is not visible due to small values of e_m^2. Values on y-axis are jittered

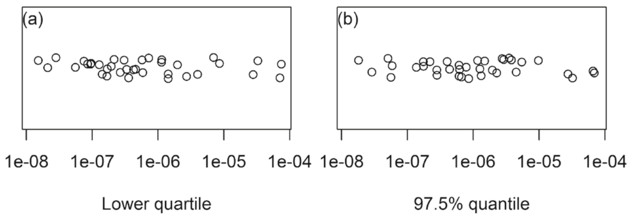

Fig. 14 Stripchart of squared Euclidean distances between posterior 25% and 97.5% quantiles of original model and Merge & Reduce model for all Bayesian Poisson regression models for M&R approach 2. x-axes are on a logarithmic scale. Vertical dashed lines at 0.1 are not visible due to small values of e_m^2. Values on y-axes are jittered

from a simulated data set with $\frac{n_b}{d} = 40$ and lead to comparatively high values of f_{se}^m, which are nevertheless well below 1.025.

M&R approach 2

To evaluate the performance of Merge & Reduce for Bayesian Poisson regression models, we first look at the squared Euclidean distances between the original posterior means and the posterior means based on Merge & Reduce. Figure 13 shows a one-dimensional scatterplot of the distances. All values are very small, and the largest squared distance lies below 10^{-4}. As in the frequentist case, the posterior location is well-recovered by Merge & Reduce.

To characterize the posterior distribution, in addition to measures of location, we again look at posterior quantiles and the posterior standard deviation. Figure 14 contains the squared Euclidean distances for the lower quartile and the 97.5% quantile as representatives. The quantiles are corrected according to Eq. 23. With the correction, all squared distances obtain very low values; again, all lie below 10^{-4}.

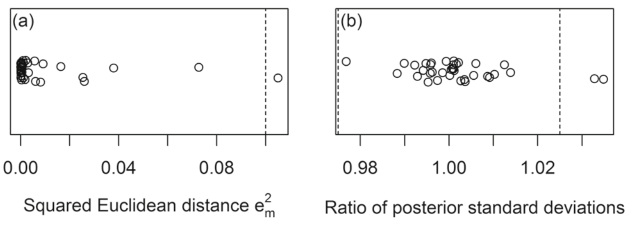

Fig. 15 Stripcharts of different measures for the ability of Merge & Reduce to recover the posterior standard deviation for M&R approach 2. **a** Squared Euclidean distance between standard deviations based on original model and Merge & Reduce model, **b** corrected factor of posterior standard deviation according to Merge & Reduce and posterior standard deviation according to original model. Values on y-axes are jittered

This picture is slightly different for the posterior standard deviation. Figure 15 shows two different measures of how close the posterior standard deviations based on Merge & Reduce are to the original models' posterior standard deviations; subfigure (a) shows the corrected factor (Eq. 22), subfigure (b) shows the squared Euclidean distance between the posterior standard deviations. The corrected factor is close to 1 for all settings of the simulation study. For two settings the ratio is above 1.025; these are the settings with the lowest values of $\frac{n_b}{d} = 20$. The squared distance between the posterior distributions shows a clear concentration close to 0. With the exception of two settings, all squared distances are below 0.05, and one of these exceptions is below 0.1. Interestingly, these relatively high values are inconspicuous with regard to the ratio. However, these are also data sets where the number of observations per block is only 400.

4 Bicycle data

We also evaluate the method on a real data set. We follow the approach taken in [24] and analyze the bicycle rental data set in [20] to that end. The data set contains information spanning two years about the number of bikes rented as well as calendrical and meteorological information on an hourly basis.

Table 2 gives an overview of all variables we consider in our models. The data set contains more variables that have to be excluded for different reasons. Some of the variables are highly correlated, e.g., the temperature and the apparent temperature. We exclude other variables, because they pose a problem when applying the Merge & Reduce technique. This is the case if the values of factor variables appear in a very systematic way, e.g., the variable *yr*, which indicates whether the observation belongs to the first or second year under study. This variable exhibits one change (from 2011

Table 2 Variables from the bike sharing data set used in the models

Variable	Description	Remark
cnt	Number of rental bikes used	Y-variable
hour	Hour (0 to 23)	Factor (24 levels)
holiday	Public holiday	Factor (2 levels)
weekday	Day of the week	Factor (7 levels)
weathersit	Weather ("clear" to "rain")	Factor (3 levels)
atemp	Apparent temperature	Standardized
hum	Humidity	Standardized
windspeed	Windspeed	Standardized

to 2012) in the middle of the data set, but otherwise stays constant. Such factor variables introduce problems with identifiability when employing Merge & Reduce as not all of the values are present for some blocks and their effect on the dependent variable can thus not be estimated. The variable *weathersit* contains four levels in the original data set. Level 4, which stands for "heavy rain", is only present 3 times out of a total of $n = 17{,}379$ observations. As in [24], we combine levels 3 ("light rain") and 4 to obtain a new level 3 ("rain"). For more detailed information about the data set and all variables, including variables not employed in our model, please refer to the description of the data set on the UCI Machine Learning Repository.[1]

In the following, we consider two sets of variables: a small subset where only the quantitative independent variables *atemp*, *hum*, and *windspeed* are included and a second, larger model that is similar to the model in [24], but excludes the variables *yr* and *season*. For the two sets of variables, we analyze both a linear and a Poisson regression model. For the linear model, the dependent variable is transformed using the logarithm.

The original data set contains a moderate number of $n = 17\,379$ observations. For that reason we only employ block sizes of $n_b \in \{400, 1000, 5000, 10{,}000\}$. Table 3 gives an overview of how close the results according to Merge & Reduce are to the original model for the frequentist analysis, while Table 4 shows the results for the Bayesian analysis.

Tables 3 and 4 indicate that the approximation of the original model using Merge & Reduce is good for block sizes of $n_b \in \{5000, 10{,}000\}$ but not good enough for $n_b \in \{400, 1000\}$, regardless of the subset of variables, for both linear and Poisson regression and for both frequentist and Bayesian models.

While the results are similar across all settings, a difference can be seen between the model including only the three quantitative variables and the model which includes factor variables as well. Especially in the Bayesian case, the model with factors shows large deviations from the original model when the block size is small. On closer inspection, this is mainly due to the unbalancedness of the variable *holiday*, which leads to blocks containing no holiday and thus to divergent models where at least some elements of the posterior distribution of β are meaningless. The problem is also present in the frequentist case, and here some elements of $\hat{\beta}$ cannot be estimated and are thus set to `NA`. We have omitted these incomparable differences from Tables 3 and 4. Notably, this illustrates the importance of careful model selection with possibly little information and indicates that n_b should not be chosen too small.

As mentioned in Sect. 3.2, [29] recommends a ratio of at least 20 observations per variable for a reliable linear model. Harrell uses the effective number of observations n_{eff} instead of n. For models that include only quantitative variables, $n_{\text{eff}} = n$, but if factor variables are included in the model, the effective number of observations changes. Let k be the number of factor levels and let n_i be the number of observations for level i, $(i = 1, \dots, k)$. The effective number of observations n_{eff} is then given by

$$n_{\text{eff}} = \min(n_1, n_2) \tag{24}$$

for binary variables and by

$$n_{\text{eff}} = n - \frac{1}{n^2} \sum_{i=1}^{k} n_i^3 \tag{25}$$

for factor variables with $k > 2$ levels [see 29][Table 4.1]. We calculate n_{eff} for all blocks and for all four values of n_b. For models that only include quantitative variables, n_{eff} is constant except for the last block. For models with factors, n_{eff} typically varies as the frequency of the different levels varies across blocks. Table 5 reports the smallest number of n_{eff} observed across all blocks with the respective value of n_b as well as the resulting minimal number of observations per variable.

From Table 5 it becomes clear that 400 and 1000 are not adequate block sizes for the bicycle rental data set when employing a model that includes factor variables. For $n_b = 400$, around half of all blocks do not contain a holiday. For $n_b = 1000$, this is only the case for one block. However, even the blocks that do contain a holiday typically include only one holiday, giving an effective number of observations of 24, which is less than the number of variables. This underlines that care should be taken especially when possibly unbalanced binary variables are present in the model.

When employing a model that only includes the three quantitative variables *atemp*, *hum*, and *windspeed* as well as an intercept term, the effective number of observations is considerably larger and the minimal number of observations

[1] http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset.

Table 3 Results of the frequentist analyses of the bicycle sharing data set

	Quantitative variables only				Including factor variables			
	logarithmic		Poisson		logarithmic		Poisson	
n_b	e_m^2	f_{se}^m	e_m^2	f_{se}^m	e_m^2	f_{se}^m	e_m^2	f_{se}^m
10,000	0.0676	1.0078	0.0443	1.0184	0.1595	0.9412	0.0988	1.0139
5000	0.0796	1.0279	0.0316	1.0574	0.0898	0.9193	0.0288	1.0182
1000	2.6690	1.4216	0.3666	1.5174				
400	5.9530	1.5078	0.9500	1.6528				

Each row headed with e^2 gives the squared Euclidean distance between the original model and the model obtained using Merge & Reduce with the given block size n_b, while each row headed with a f_{se} shows the corrected standard error factor. A total of four models are analyzed: two models using only quantitative variables as independent variables and two models including factor variables as well. For each of these models, a linear regression (using a logarithmic transformation of the number of shared bikes as dependent variable) and a Poisson regression analysis are conducted

Table 4 Euclidean squared distances e_m^2 r for the Bayesian analyses of the bicycle sharing data set

Variab.	Model	n_b	$\tilde{x}_{0.025}$	$\tilde{x}_{0.25}$	$\bar{x}$	$\tilde{x}_{0.5}$	$\tilde{x}_{0.75}$	$\tilde{x}_{0.975}$	s
Quant.	Log.	10,000	0.0683	0.0683	0.0678	0.0676	0.0672	0.0665	1.0016
Quant.	Log.	5000	0.0783	0.0792	0.0810	0.0809	0.0834	0.0872	1.0148
Quant.	Log.	1000	2.4053	2.5722	2.6725	2.6761	2.7800	2.9838	1.3290
Quant.	Log.	400	5.4856	5.7779	5.9472	5.9437	6.1249	6.4711	1.3973
Quant.	Pois.	10,000	0.0445	0.0443	0.0443	0.0443	0.0442	0.0442	1.0262
Quant.	Pois.	5000	0.0315	0.0315	0.0316	0.0316	0.0317	0.0317	1.0634
Quant.	Pois.	1000	0.3592	0.3641	0.3667	0.3667	0.3694	0.3742	1.5349
Quant.	Pois.	400	0.9368	0.9455	0.9502	0.9503	0.9549	0.9635	1.6705
Factors	Log.	10,000	0.1674	0.1636	0.1619	0.1622	0.1608	0.1579	0.9298
Factors	Log.	5000	0.0999	0.0953	0.0932	0.0933	0.0915	0.0873	0.9128
Factors	Pois.	10,000	0.0988	0.0988	0.0989	0.0988	0.0989	0.0991	1.0208
Factors	Pois.	5000	0.0291	0.0289	0.0288	0.0288	0.0287	0.0286	1.0141

A total of four models are analyzed: two models using only quantitative variables as independent variables ("quant.") and two models including factor variables as well ("factors"). For each of these models, a linear regression (using a logarithmic transformation of the number of shared bikes as dependent variable) and a Poisson regression analysis are conducted. Every row shows the results for one of the four models and the respective block size n_b. Given are the approximations of different posterior quantiles $\tilde{x}$ as well as posterior mean and standard deviation. The models with factor variables and block sizes $n_b \leq 1000$ did not converge, resulting in meaningless large deviations from the original model (not shown)

Table 5 Smallest value of effective number of observations n_{eff} for different block sizes n_b as well as minimal effective number of observations divided by number of parameters d, where $d = 36$ including all dummy variables

Block size n_b	Quantitative variables only		Including factor variables	
	$\min n_{\text{eff}}$	$\frac{\min n_{\text{eff}}}{d}$	$\min n_{\text{eff}}$	$\frac{\min n_{\text{eff}}}{d}$
400	179	44.75	0	0
1000	379	94.75	0	0
5000	2379	594.75	95	2.64
10,000	7379	1844.75	191	5.31

per variable is 44.75 even for a block size of $n_b = 400$. Despite these relatively high numbers, the approximation of the approximation is only acceptable for block sizes 5000 and 10,000.

The difference between the model with the three quantitative variables and the model including factor variables is a considerable discrepancy in goodness of fit. To name just one example, the time of day plays an important role for the number of bicycles rented, but this variable is not included in the smaller model.

The results of our analysis indicate that both the number of observations per block per variable and the model's goodness of fit play important roles for Merge & Reduce to deliver good approximations.

5 Conclusions

In this article, we introduced the general algorithmic scheme of Merge & Reduce and developed conceptually new schemes acting directly on statistical models. This enabled us to perform computationally demanding statistical data analysis tasks on massive data streams with n observations. The data are divided into small tractable batches from which we compute individual models using offline state of the art or classical data analysis algorithms. The resulting models are combined in a structured way, such that blow-up in their memory requirements remains bounded by a negligible $O(\log n)$ factor. It is only necessary to choose an appropriate statistical model and implement merge and reduce operations for the specific type of model. The design of such operations is not trivial in general.

In particular we showed how to design such operations for the case of (Bayesian) linear regression, Gaussian mixture models and generalized linear models. We evaluated our M&R approaches on simulated and real-world data. The Merge & Reduce algorithm performed well, leading to stable results very similar to the models one would obtain from analyzing the entire data set as a whole.

In the case where the data consist of a mixture of two different locations within X and Y, models that show a clear break point—as opposed to a homogeneous fraction of outliers in each block—are more difficult to recover for Merge & Reduce, especially if the model includes an intercept term. In such a situation, employing approaches like tests for structural breaks [55] or equivalence testing [17,34] for regression models might be beneficial. These procedures typically require access to the whole data set at once, making a transfer to the Merge & Reduce framework, possibly in combination with a suitable measure for the goodness of fit, an interesting open problem.

M&R approach 3 is suitable for frequentist linear regression models and is able to recover the original model exactly. M&R approaches 1 and 2 are suitable for a variety of frequentist and Bayesian regression models, respectively. For both approaches, the goodness of the approximation depends on both the ratio of observations per block and variables $\frac{n_b}{d}$ and the goodness-of-fit of the original model. The first condition can easily be controlled by the data analyst, especially in a setting with large n. The second condition may require care when building the model. Both conditions are also present when employing regular regression analysis, albeit in slightly weaker form. Future work may focus on designing and extending the statistical models tractable in the Merge & Reduce framework to classification [41,45], clustering [49] and topic modeling [7].

For massively large data not fitting into internal memory, analyzing the entire data set becomes tedious or may not be possible at all, whereas the Merge & Reduce scheme enables the data analysis by performing the computations on small, efficiently tractable batches. The same is true for analyzing moderate amounts of data on severely resource-restricted computational devices like mobile phones or mobile sensors. Additionally we pointed out that the scheme is also applicable in parallel or distributed settings. This underlines the versatility of the Merge & Reduce scheme beyond classical statistical data analysis.

Acknowledgements Open Access funding provided by Projekt DEAL. This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", Project C4, and by the Dortmund Data Science Center (DoDSc).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Agarwal, P.K., Sharathkumar, R.: Streaming algorithms for extent problems in high dimensions. Algorithmica **72**(1), 83–98 (2015)
2. Agarwal, P.K., Har-Peled, S., Varadarajan, K.R.: Approximating extent measures of points. J. ACM **51**(4), 606–635 (2004)
3. Badoiu, M., Clarkson, K.L.: Smaller core-sets for balls. In: Proceedings of SODA, pp. 801–802 (2003)
4. Badoiu, M., Clarkson, K.L.: Optimal core-sets for balls. Comput. Geom. **40**(1), 14–22 (2008)
5. Badoiu, M., Har-Peled, S., Indyk, P.: Approximate clustering via core-sets. In: Proceedings of STOC, pp. 250–257 (2002)
6. Balakrishnan, S., Madigan, D.: A one-pass sequential Monte Carlo method for Bayesian analysis of massive datasets. Bayesian Anal. **1**(2), 345–361 (2006)
7. Bansal, T., Bhattacharyya, C., Kannan, R.: A provable SVD-based algorithm for learning topics in dominant admixture corpus. In: Proceedings of NeurIPS, pp. 1997–2005 (2014)
8. Bentley, J.L., Saxe, J.B.: Decomposable searching problems I: static-to-dynamic transformation. J. Algorithms **1**(4), 301–358 (1980)
9. Bruno, N., Chaudhuri, S.: Physical design refinement: the 'merge-reduce' approach. ACM Trans. Database Syst. **32**(4), 28 (2007)
10. Clarkson, K.L.: Subgradient and sampling algorithms for ℓ_1 regression. In: Proceedings of SODA, pp. 257–266 (2005)

11. Clarkson, K.L., Woodruff, D.P.: Input sparsity and hardness for robust subspace approximation. In: Proceedings of FOCS, pp. 310–329 (2015)
12. Clarkson, K.L., Woodruff, D.P.: Sketching for M-estimators: a unified approach to robust regression. In: Proceedings of SODA, pp. 921–939 (2015)
13. Clarkson, K.L., Drineas, P., Magdon-Ismail, M., Mahoney, M.W., Meng, X., Woodruff, D.P.: The fast Cauchy transform and faster robust linear regression. SIAM J. Comput. **45**(3), 763–810 (2016)
14. Cohen, M.B., Lee, Y.T., Musco, C., Musco, C., Peng, R., Sidford, A.: Uniform sampling for matrix approximation. In: Proceedings of ITCS, pp. 181–190 (2015)
15. Dasgupta, A., Drineas, P., Harb, B., Kumar, R., Mahoney, M.W.: Sampling algorithms and coresets for ℓ_p regression. SIAM J. Comput. **38**(5), 2060–2078 (2009)
16. Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Commun. ACM **53**(1), 72–77 (2010)
17. Dette, H., Möllenhoff, K., Volgushev, S., Bretz, F.: Equivalence of regression curves. J. Am. Stat. Assoc. **113**(522), 711–729 (2018). https://doi.org/10.1080/01621459.2017.1281813
18. Drineas, P., Mahoney, M.W., Muthukrishnan, S.: Sampling algorithms for ℓ_2 regression and applications. In: Proceedings of SODA, pp. 1127–1136 (2006)
19. Drineas, P., Mahoney, M.W., Muthukrishnan, S.: Relative-error CUR matrix decompositions. SIAM J. Matrix Anal. Appl. **30**(2), 844–881 (2008)
20. Fanaee-T, H., Gama, J.: Event labeling combining ensemble detectors and background knowledge. Prog. AI **2**(2–3), 113–127 (2014)
21. Feldman, D., Faulkner, M., Krause, A.: Scalable training of mixture models via coresets. In: Proceedings of NeurIPS, pp. 2142–2150 (2011)
22. Feldman, D., Schmidt, M., Sohler, C.: Turning Big Data into tiny data: constant-size coresets for k-means, PCA and projective clustering. In: Proceedings of SODA, pp. 1434–1453 (2013)
23. Feldman, D., Munteanu, A., Sohler, C.: Smallest enclosing ball for probabilistic data. In: Proceedings of SoCG, pp. 214–223 (2014)
24. Geppert, L.N., Ickstadt, K., Munteanu, A., Quedenfeld, J., Sohler, C.: Random projections for Bayesian regression. Stat. Comput. **27**(1), 79–101 (2017)
25. Groß, J.: Linear Regression. Springer, Berlin (2003)
26. Har-Peled, S.: A simple algorithm for maximum margin classification, revisited. CoRR abs/1507.01563 (2015)
27. Har-Peled, S., Mazumdar, S.: On coresets for k-means and k-median clustering. In: Proceedings of STOC, pp. 291–300 (2004)
28. Har-Peled, S., Roth, D., Zimak, D.: Maximum margin coresets for active and noise tolerant learning. In: Proceedings of IJCAI, pp. 836–841 (2007)
29. Harrell Jr., F.E.: Regression Modeling Strategies. Springer, New York (2001)
30. Hoffman, M.D., Gelman, A.: The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. **15**(1), 1593–1623 (2014)
31. Huggins, J.H., Campbell, T., Broderick, T.: Coresets for scalable Bayesian logistic regression. In: Proceedings of NeurIPS, pp. 4080–4088 (2016)
32. Law, J., Wilkinson, D.J.: Composable models for online Bayesian analysis of streaming data. Stat. Comput. **28**(6), 1119–1137 (2018)
33. Li, M., Miller, G.L., Peng, R.: Iterative row sampling. In: Proceedings of FOCS, pp. 127–136 (2013)
34. Liu, W., Bretz, F., Hayter, A.J., Wynn, H.P.: Assessing nonsuperiority, noninferiority, or equivalence when comparing two regression models over a restricted covariate region. Biometrics **65**(4), 1279–1287 (2009). https://doi.org/10.1111/j.1541-0420.2008.01192.x
35. Lucic, M., Bachem, O., Krause, A.: Strong coresets for hard and soft Bregman clustering with applications to exponential family mixtures. In: Proceedings of AISTATS, pp. 1–9 (2016)
36. McCullagh, P., Nelder, J.A.: Generalized Linear Models. Chapman & Hall, London (1989)
37. Molina, A., Munteanu, A., Kersting, K.: Core dependency networks. In: Proceedings of AAAI (2018)
38. Munteanu, A., Schwiegelshohn, C.: Coresets-methods and history: a theoreticians design pattern for approximation and streaming algorithms. KI **32**(1), 37–53 (2018)
39. Munteanu, A., Schwiegelshohn, C., Sohler, C., Woodruff, D.P.: On coresets for logistic regression. In: Proceedings of NeurIPS, pp. 6562–6571 (2018)
40. Muthukrishnan, S.: Data streams: algorithms and applications. Found Trends Theor. Comput. Sci. **1**(2), 117–236 (2005)
41. Naik, A., Rangwala, H.: Hierflat: flattened hierarchies for improving top-down hierarchical classification. Int. J. Data Sci. Anal. **4**(3), 191–208 (2017)
42. Phillips, J.M.: Coresets and sketches. In: Goodman, J.E., O'Rourke, J., Tóth, C.D. (eds.) Handbook of Discrete and Computational Geometry, 3rd edn, pp. 1269–1288. CRC, Boca Raton (2017)
43. R Core Team.: R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org (2014, 2017, 2018)
44. Reddi, S.J., Póczos, B., Smola, A.J.: Communication efficient coresets for empirical loss minimization. In: Proceedings of UAI, pp. 752–761 (2015)
45. Sambasivan, R., Das, S.: Classification and regression using augmented trees. Int. J. Data Sci. Anal. **7**(4), 259–276 (2019)
46. Sohler, C., Woodruff, D.P.: Subspace embeddings for the L_1-norm with applications. In: Proceedings of STOC, pp. 755–764 (2011)
47. Sokolova, M.: Big text advantages and challenges: classification perspective. Int. J. Data Sci. Anal. **5**(1), 1–10 (2018)
48. Stan Development Team.: RStan: the R interface to Stan. http://mc-stan.org/. R package version 2.14.1 (2016)
49. Teffer, D., Srinivasan, R., Ghosh, J.: Adahash: hashing-based scalable, adaptive hierarchical clustering of streaming data on mapreduce frameworks. Int. J. Data Sci. Anal. **8**(3), 257–267 (2019)
50. Tolochinsky, E., Feldman, D.: Coresets for monotonic functions with applications to deep learning. CoRR abs/1802.07382 (2018)
51. Weihs, C., Ickstadt, K.: Data science: the impact of statistics. Int. J. Data Sci. Anal. **6**(3), 189–194 (2018)
52. Welling, M., Teh, Y.W., Andrieu, C., Kominiarczuk, J., Meeds, T., Shahbaba, B., Vollmer, S.: Bayesian inference & big data: a snapshot from a workshop. ISBA Bull. **21**(4), 8–11 (2014)
53. Woodruff, D.P.: Sketching as a tool for numerical linear algebra. Found Trends Theor. Comput. Sci. **10**(1–2), 1–157 (2014)
54. Woodruff, D.P., Zhang, Q.: Subspace embeddings and ℓ_p-regression using exponential random variables. In: Proceedings of COLT, pp. 546–567 (2013)
55. Zeileis, A., Kleiber, C., Krämer, W., Hornik, K.: Testing and dating of structural changes in practice. Comput. Stat. Data Anal. **44**(1–2), 109–123 (2003). https://doi.org/10.1016/s0167-9473(03)00030-6