Salloum *et al. J Big Data* (2019) 6:45
https://doi.org/10.1186/s40537-019-0205-4

Journal of Big Data

RESEARCH Open Access

Exploring and cleaning big data with random sample data blocks

Salman Salloum[1,2], Joshua Zhexue Huang[1,2] and Yulin He[1,2*]

*Correspondence:
yulinhe@szu.edu.cn
[1] Big Data Institute, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
Full list of author information is available at the end of the article

Abstract

Data scientists need scalable methods to explore and clean big data before applying advanced data analysis and mining algorithms. In this paper, we propose the RSP-Explore method to enable data scientists to iteratively explore big data on small computing clusters. We address three main tasks: statistical estimation, error detection, and data cleaning. The Random Sample Partition (RSP) distributed data model is used to represent the data as a set of ready-to-use random sample data blocks (called RSP blocks) of the entire data. Block-level samples of RSP blocks are selected to understand the data, identify potential types of value errors, and get samples of clean data. We provide a theoretical analysis on using RSP blocks for statistical estimation and demonstrate empirically the advantages of the RSP-Explore method. The experimental results of three real data sets show that the approximate results from RSP-Explore can rapidly converge toward the true values. Furthermore, cleaning a sample of RSP blocks is sufficient to estimate the statistical properties of the unknown clean data.

Keywords: Big data, Exploratory data analysis, Statistical estimation, Data cleaning, Block-level sampling, Random sample partition, Distributed, Parallel and cluster computing

Introduction

Motivation

Sampling-based approaches have been adopted to alleviate the burden of big data volume not only when approximate results are useful as exact ones [1–5], but also when the results from a small clean sample can be more accurate than those from the entire dirty data [6–9]. It is a common practice to iteratively generate small random samples of a big data set to explore the statistical properties of the entire data and define cleaning rules [10–19]. This iterative process becomes impractical or impossible on small computing clusters due to the communication, I/O and memory costs of cluster computing frameworks that implement a shared-nothing architecture [20–22]. While these distributed frameworks have not adapted well to the requirements of data exploration tasks, existing sequential techniques don't scale easily to big data [23]. In fact, there are plenty of data exploration and analysis libraries in common data science languages, e.g., R and Python [24, 25]. To scale these libraries to big data on computing clusters, new distributed implementations are required to process distributed data. Even with distributed algorithms, the memory of the computing cluster may not be enough to hold the entire

data. This is because the data growth rate is outpacing the technology scaling [26]. In this paper, our objective is enabling data scientists to iteratively explore and clean big data on small computing clusters by applying existing or user-defined algorithms to a few ready-to-use random sample data blocks.

On computing clusters, big data is divided into small disjoint distributed data blocks in Hadoop Distributed File System (HDFS) [27]. In fact, HDFS blocks are the basic units of both storage and processing in cluster computing frameworks (e.g., Apache Hadoop[1], Apache Spark[2]). The MapReduce computing model [28] is adapted to process HDFS blocks in parallel and get the result for the entire data set [29–31]. Hadoop-based computing clusters have become the norm for big data management and analysis in different application areas [32–36]. To facilitate random sampling on these clusters, the *Random Sample Partition (RSP)* distributed data model was proposed recently [37]. In this model, the statistical properties of the data set are preserved in its distributed data blocks by making HDFS blocks as *ready-to-use random samples (RSP blocks)* of the entire data. This can avoid both the cost of record-level sampling (RLS) and the biased results from Block-Level Sampling (BLS) of inconsistent HDFS blocks [38–41]. An RSP is generated offline, and only once, from an HDFS file using a two-stage data partitioning method [42, 43]. To analyze the data set, a block-level sample of RSP blocks is selected and processed in parallel using a sequential algorithm to get an approximate result for the entire data set. The RSP approach enables approximate big data analysis on small computing clusters especially when the results from a few RSP blocks are equivalent to those from the entire data set [3, 44].

Contributions

Given the statistical and computational advantages of the RSP approach, we employ this approach to address the problem of big data exploration and cleaning on small computing clusters. In this paper, we propose RSP-Explore, an RSP-based method to explore, validate and clean big data using a few RSP blocks. Given an RSP stored on a computing cluster, this method enables data scientists to estimate the statistical properties of the entire data set while tuning the amount of processed data according to the available resources. Block-level samples of RSP blocks are used to tackle three main tasks: statistical estimation, error detection, and data cleaning. Since RSP blocks are random samples of the same size, the basic principle in this method is using sample estimates from individual RSP blocks as an approximation of the sampling distribution of the sample estimate. From this sampling distribution, an approximate result for the entire data is obtained with a confidence interval. This principle is employed to estimate the summary statistics of single features and the correlation coefficients between features. Similarly, the error detection problem is addressed by estimating the sampling distribution of the sample proportion of errors, outliers, missing, and valid values from a sample of RSP blocks.

[1] http://hadoop.apache.org/.

[2] https://spark.apache.org/.

With RSP-Explore, a preliminarily understanding of the general statistical properties of the data and the potential types of inconsistent values can be obtained earlier and without computing the entire data set. The estimated statistics and proportions guide the definition of data cleaning rules. These rules are applied in parallel to clean samples of RSP blocks. From the cleaned RSP blocks, summary statistics of the unknown clean data are estimated. The three operations, i.e., estimation, detection, and cleaning, can be repeated either independently or in a data pipeline to improve the results. The experimental results of three real data sets show that estimates from a sample of RSP blocks can rapidly converge toward the true values and that cleaning a sample of dirty RSP blocks is enough to estimate the statistical properties of the unknown clean data.

The main contributions of this paper are as follows:

- We propose RSP-Explore, a new method for big data exploration and cleaning on computing clusters using the RSP approach.
- We address error detection as an estimation problem by estimating the proportions of inconsistent values in quantitative data;
- We propose an algorithm to estimate the statistical properties of the entire unknown clean data set by cleaning only a few RSP blocks;
- We introduce a theoretical analysis on using RSP blocks for statistical estimation;
- We empirically demonstrate the performance of RSP-Explore on three real data sets.

The remainder of this paper is organized as follows. In "Related work" section, we briefly review related work. Then, we describe the Random Sample Partition (RSP) approach in "Background" section. After that, we propose the RSP-Explore method in "Methods" section. We demonstrate the performance of this method on small computing cluster in "Results" section and discuss the implications of this method in "Discussion" section. Finally, we conclude this paper in "Conclusions" section.

Related work

In this section, we briefly review recent research directions on exploring and cleaning big data on computing clusters and show the differences from our method.

Sampling-based approximate big data analysis

Sampling-based approximation has become a common approach for big data analysis in cluster computing frameworks. ApproxHadoop [2] uses online multi-stage sampling from HDFS blocks to get approximate results. BlinkDB [45] is a distributed approximate query processing engine that uses offline stratified sampling on frequently occurring columns and uniform sampling to support ad-hoc queries. IncApprox [46] uses online stratified sampling to produce an incrementally updated approximate result from streaming data. These frameworks operate directly on the data and compute error bounds without considering data errors. SampleClean [6], on the other hand, combines sampling and cleaning to improve accuracy of aggregate queries. A random sample of dirty data is created first, and then a data cleaning technique is applied to clean the sample. Next, the cleaned sample is used to estimate the results of aggregate queries. SampleClean supports closed-form estimates based on normal approximation. A query is formulated

first as calculating the mean value so that the confidence interval of the result can be estimated according to the Central Limit Theorem. ActiveClean [9] is an iterative data cleaning framework that cleans a small sample of the data to produce a model similar to if the full data set were cleaned. It starts with a dirty model, then incrementally cleans a new sample and updates the model. ActiveClean supports convex loss models and uses the model as guide to identify future data to clean by cleaning those records likely to affect the results.

RSP-Explore has several key differences from these works. First, the entire data is stored as ready-to-use disjoint random sample data blocks using the RSP distributed data model. Second, RSP-Explore targets at offline workloads where data scientists use a variety of techniques to infer global statistical properties from large data volumes. Third, RSP-Explore quantifies the uncertainty of the estimated result using the sampling distribution rather than closed-form approximation. Thus, estimators can be used directly without manual reformulation. RSP-Explore can also be applied to predictive modeling tasks where an ensemble model is incrementally built from samples of clean RSP blocks.

Big data exploration and profiling

There are a number of works on extending the mainstream cluster computing frameworks for exploratory data analysis. Optimus[3] implements common data exploration and cleaning operations on Spark DataFrames. Spark DataFrame Profile[4] generates statistics (e.g., descriptive statistics, quantiles, histogram) from Spark DataFrames. Cumulon [47] is an end-to-end system to optimize the cost of calculating statistics on the cloud. Sketch [48] is a system for aggregation on distributed data sets. Data Canopy [49] computes and maintains basic statistics (mean, variance, standard deviation, correlation, and covariance) from fixed chunks of data. *While these frameworks apply operations to the entire data set, RSP-Explore applies sequential functions to a few RSP blocks without loading the entire data. RSP-Explore can be implemented as an extension to these frameworks and the functionalities in these frameworks can be applied to explore and clean RSP blocks. For instance, a Spark Dataframe can be created from a sample of RSP blocks and processed directly with the operations in these frameworks.*

Scaling statistical methods to big data

Cluster computing frameworks with a shared-nothing architecture have been adopted to scale iterative algorithms to big data [50]. In addition, to enable data scientists to scale existing statistical methods to big data on computing clusters, libraries were developed in common data science languages such as R and Python. Dask[5] extends common interfaces in Pyhton like NumPy, Pandas to distributed environments. Ray DataFrames [23] is a library for exploratory data analysis by using the same Panda API in Python to run jobs on distributed data sets. RHIPE[6] applies the Divide-and-Recombine approach [51] to reveal deeper insights from distributed data sets by applying R statistical functions in

[3] https://hioptimus.com/.

[4] https://github.com/julioasotodv/spark-df-profiling.

[5] https://dask.pydata.org/.

[6] http://www.deltarho.org.

parallel to individual distributed data blocks. *However, the exploration and combination of the block-level results becomes a challenge to data scientists due to the large number of distributed data blocks in a big data set and the inconsistency of data distributions in HDFS blocks. In RSP-Explore, we solve this problem by first representing the data set as an RSP where each block is a random sample, and then using block-level sampling to process only a few RSP blocks. On the other hand, RSP blocks can be used directly as subsamples in statistical estimation procedures such as the Bag of Little Bootstraps (BLB)* [52].

Background

RSP is a new approach that makes the distributed data blocks of a big data set as ready-to-use random samples for approximate big data analysis [37]. This approach can be applied to different scenarios where the entire big data set can't be computed e.g., when the data volume is bigger than the available memory, or the data is stored in multiple data centers or generated in different time windows. In this section, we briefly present the RSP distributed data model and show how this model enables big data computing on small computing clusters.

RSP distributed data model

Assume that $\mathbb{D}$ is a multivariate data set of N records and M features where N is very large so $\mathbb{D}$ cannot be analyzed on a single machine. With the RSP model, $\mathbb{D}$ is divided into K small disjoint random sample data blocks in advance on a computing cluster, called RSP blocks. Assume $\mathbb{F}(x)$ is the sample distribution function (SDF) of a random variable x in $\mathbb{D}$. $\mathrm{T} = \{\mathrm{D}_1, \mathrm{D}_2, \ldots, \mathrm{D}_K\}$ is a random sample partition of $\mathbb{D}$, with K RSP blocks each has n records, if

$$E[F_k(x)] = \mathbb{F}(x) \quad \text{for} \quad \text{each} \quad k = 1, 2, \ldots, K,$$

where $F_k(x)$ denotes the sample distribution function of x in $\mathrm{D_k}$ and $E[F_k(x)]$ denotes its expectation. In practice, $\mathbb{D}$ is often stored in HDFS. An RSP is generated from an HDFS file of $\mathbb{D}$ using the two-stage data partitioning algorithm [42, 43]. Each RSP block is created by combining approximately equal random slices from all the original HDFS blocks. This operation can be scheduled to run offline on the computing cluster, i.e., before starting a data analysis session. T is saved as an HDFS-RSP file with metadata $\mathrm{T_{metadata}}$ storing RSP block information including the size and location. Selecting an RSP block from T is equivalent to drawing a random sample directly from $\mathbb{D}$. Consequently, data scientists can use RSP blocks directly in sampling-based approximate big data analysis.

Big data computing with RSP blocks

To analyze $\mathbb{D}$ with a function f on a computing cluster, only a few RSP blocks from T are randomly selected and computed in parallel on their nodes using a sequential implementation of f. This enables small computing clusters to handle data bigger than the memory size by using a step-wise process known as the *asymptotic ensemble learning process*. This process depends on $\mathrm{T_{metadata}}$ and works as follows:

1. A block-level sample with g RSP blocks, $\mathrm{S} = \left\{\mathrm{D}_{\tau_1}, \mathrm{D}_{\tau_2}, \ldots, \mathrm{D}_{\tau_g}\right\}$ is selected from T without replacement. $\tau_1, \tau_2, \ldots, \tau_g$ are the identifiers of the selected RSP blocks;

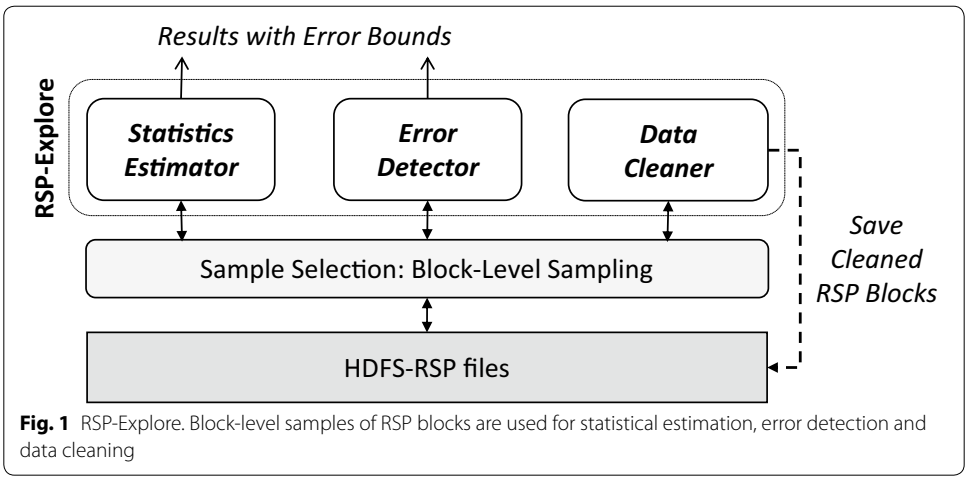

Fig. 1 RSP-Explore. Block-level samples of RSP blocks are used for statistical estimation, error detection and data cleaning

2. f is applied in parallel to each RSP block in S;
3. The outputs from these blocks are combined to produce an approximate result for $\mathbb{D}$.

To incrementally improve the results, the previous three steps can be repeated where a new block-level sample of T is used each time and the outputs from all the processed RSP blocks are combined in an approximate result for the entire data. This step-wise process can run until a satisfactory result is obtained or all RSP blocks are used up. The RSP approach was used for predictive modeling tasks such as classification and regression [44]. An RSP-based ensemble model built from a sample of RSP blocks is equivalent to a single model built from the entire data. In this paper, we employ the RSP approach for big data exploration and cleaning.

Methods

Building on the statistical and computational advantages of the RSP approach, we propose to use this approach to address the problem of big data exploration and cleaning on small computing clusters. In this section, we first introduce the RSP-Explore method to explore the statistical properties of a big data set $\mathbb{D}$ and handle data errors using samples of RSP blocks. Then, we present a theoretical analysis on using RSP blocks for statistical estimation.

RSP-Explore

Given a random sample partition T of a big data set $\mathbb{D}$ with K RSP blocks stored as an HDFS-RSP file, the RSP-Explore method enables data scientists to quickly explore the general statistical properties of $\mathbb{D}$ and define rules for data validation and cleaning. As shown in Fig. 1, this method uses block-level samples from an HDFS-RSP file to address three main tasks: statistical estimation, error detection, and data cleaning.

The Statistics Estimator is used to compute an estimate of a statistic of $\mathbb{D}$ within a confidence interval. The Error Detector is used for data validation by estimating the proportions of errors, outliers, missing and valid values in $\mathbb{D}$ within a confidence interval. In case of inconsistent values, a sample of dirty RSP blocks is cleaned with the Data Cleaner in order to estimate the statistical properties of the unknown clean data set $\mathbb{D}_{clean}$.

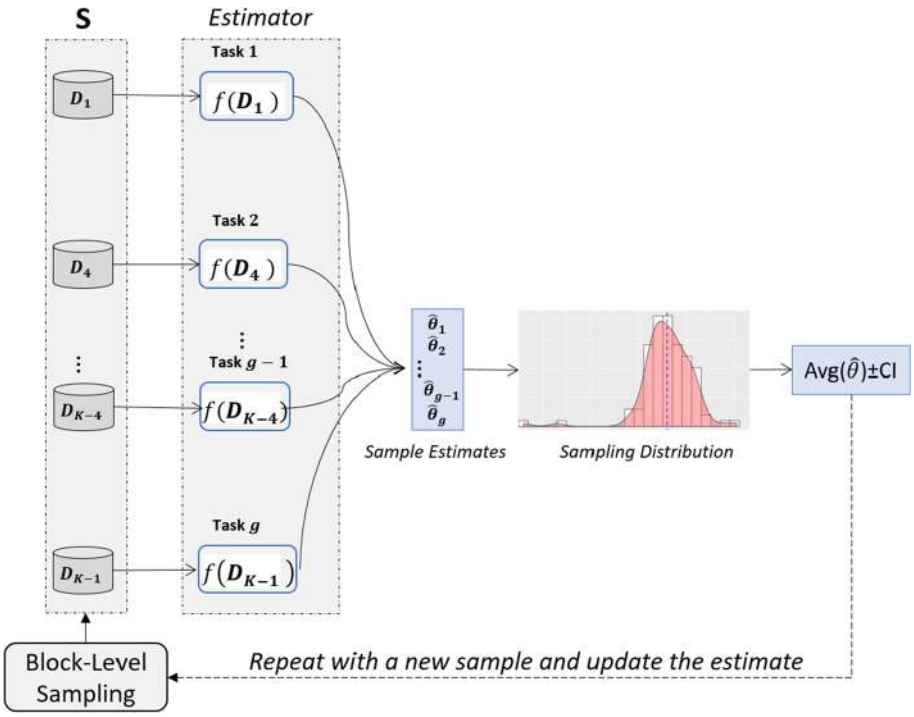

Fig. 2 Statistical estimation. Sample estimates from block-level samples of RSP blocks are computed in parallel and used as an approximate sampling distribution

Statistics estimator

Let f be an estimator function to compute an estimate $\hat{\theta}$ of a statistic θ of $\mathbb{D}$. f is used to compute a sample estimate from an RSP block. From a block-level sample S, a set $\{\hat{\theta}_q\}_{q=1}^{g}$ of g sample estimates are computed in-parallel as shown in Fig. 2. The sample estimate $\hat{\theta}_q$ is a random variable since it depends on a particular RSP block D_{τ_q}, i.e., a particular random sample. Thus, the sample estimates $\{\hat{\theta}_q\}_{q=1}^{g}$ are used as an approximation of the sampling distribution of $\hat{\theta}$. With this sampling distribution, the variability of the sample estimate can be measured and an interval estimate is calculated as an approximate result for the entire data. We apply this estimation method to univariate (mean, standard deviation, median, MAD, skewness, kurtosis) and bivariate estimators (correlation). For these estimators, the mean of the sampling distribution from S is used an approximate estimate $\hat{\theta}_{\mathrm{S}} = \frac{1}{g}\sum_{q=1}^{g}\hat{\theta}_q$. The Statistics Estimator helps data scientists understand the global statistical properties of the data (e.g., mean, standard deviation, median, MAD, skewness, kurtosis, and correlation) and define validation rules to explore potential quality issues in the data with the Error Detector.

Error detector

We address the error detection task in quantitative data as an estimation problem. Let $X = \{x_1, x_2, \ldots, x_N\}$ be a random variable defined over a domain of values $\mathbb{X}$ and represented as a feature in $\mathbb{D}$. We categorize the values of X in four categories: error, missing, outliers, and valid values:

- Error values: values that don't belong to the data type of X (e.g., a negative value in a power consumption feature);
- Outliers values: any value that is not error and located more than a specific threshold away from the center of the data. Since the mean and standard deviation are sensitive to outliers, we use the median as a robust metric of location instead of the mean, and the Median Absolute Deviation (MAD) as a robust metric of dispersion instead of the standard deviation [16, 53]. The MAD measures the median distance of all the values from the median. The outliers threshold is then defined as $a \times MAD$ away from the median where a is often set to 2, 2.5, or 3.
- Missing values are often represented using a special string, e.g., NA;
- Valid values don't belong to any of the previous categories.

The proportion of each category is estimated in a similar way to estimating the statistics of $\mathbb{D}$. Let h be a predicate function such that $h(x_i) = 1$ wherever X satisfies a given predicate, corresponding to one of the previous categories, and $h(x_i) = 0$ otherwise for $1 \leq i \leq N$. The estimator function f is defined to calculate the proportion of values that satisfy the target predicate. The proportion of values of X that satisfy the predicate in $\mathbb{D}$ is $p = \frac{1}{N} \sum_{i=1}^{N} (h(x_i) = 1)$ and the proportion of values that satisfy the predicate in an RSP block D_{τ_q} is $\hat{p}_q = \frac{1}{n} \sum_{j=1}^{n} (h(x_j) = 1)$ for $1 \leq q \leq g$. From a block-level sample S, g sample proportions are calculated in-parallel and used as an approximation of the sampling distribution of the sample proportion. Then, the proportion of values that satisfy the predicate in S is $\hat{p}_{\mathrm{S}} = \frac{1}{g} \sum_{q=1}^{g} \hat{p}_q$. This average value is returned as an approximate proportion of the true proportion p with a confidence interval. For each of the four categories of values, a predicate is defined to get the approximate proportion from a sample of RSP blocks. The Error Detector can alleviate a critical issue when analyzing big data by estimating the proportion of inconsistent values without computing the entire data or directly running expensive cleaning operations. The estimated proportions guide the definition of rules to clean the data with the Data Cleaner.

Data cleaner

Let C be a function that applies transformation and imputation operations to repair inconsistent values. In practice, C can be any data cleaning technique. A cleaning rule is defined for each category of the inconsistent values. It determines whether to delete or replace the inconsistent values and what the replacing value is, e.g., mean or median. The Data Cleaner enables data scientists to apply C in parallel to each RSP block in S and produce a set of cleaned RSP blocks $\mathrm{S}_{\mathrm{clean}} = \left\{\mathrm{D'}_{\tau_1}, \mathrm{D'}_{\tau_2}, \ldots, \mathrm{D'}_{\tau_g}\right\}$. These cleaned RSP blocks can be saved in HDFS and the metadata of their locations are recorded in $\mathrm{T}_{\mathrm{metadata}}$. For each RSP block, the cleaning status and the location of the cleaned block are maintained so that these blocks can be used to estimate the statistical properties of the unknown clean data set $\mathbb{D}_{clean}$ with the Statistics Estimator.

As RSP blocks of a clean big data set $\mathbb{D}$ are random samples of $\mathbb{D}$, cleaned RSP blocks of $\mathbb{D}$ are potential representative samples of $\mathbb{D}_{clean}$. In this paper, we argue that cleaning a sample of dirty RSP blocks from T is sufficient to estimate the statistical properties of the unknown clean data set $\mathbb{D}_{clean}$. Instead of cleaning the entire data, which is often not necessary for data exploration, the Data Cleaner and Statistics Estimator are used in a pipeline to explore the

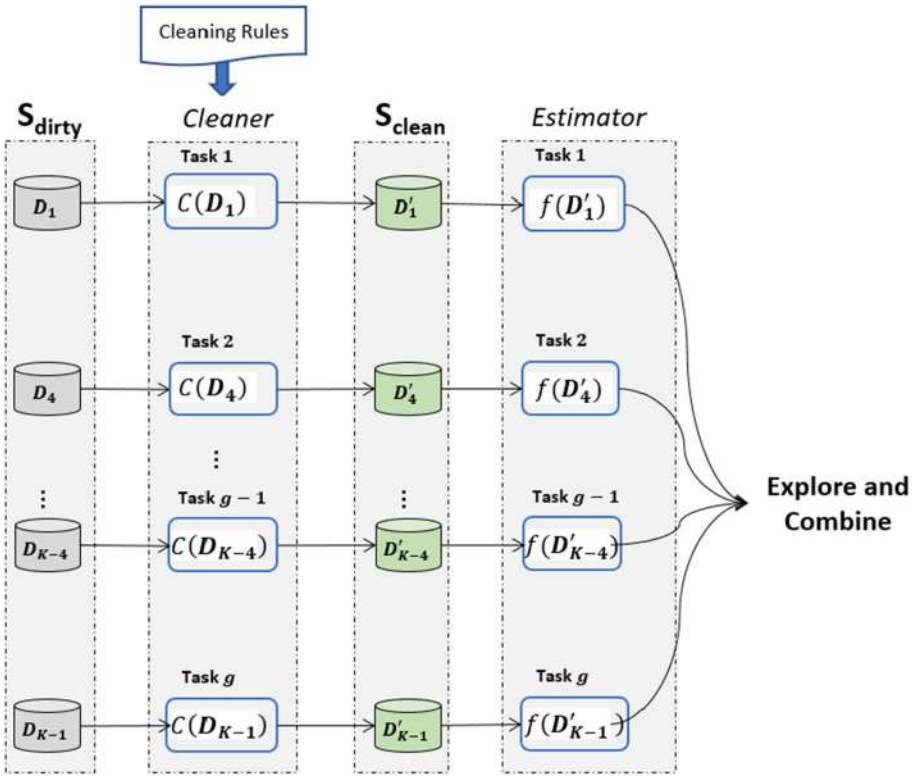

Fig. 3 Cleaning and estimation. Cleaning block-level samples of dirty RSP blocks to estimate statistics of the entire unknown clean data

statistical properties of $\mathbb{D}_{clean}$ as shown in Fig. 3. Each RSP block $\mathrm{D_q}$ is processed first with C to get a clean RSP block $\mathrm{D'_{\tau_q}}$ for $1 \leq q \leq g$. Then, f is applied to get an estimate $\hat{\theta}'_q$ from $\mathrm{D'_{\tau_q}}$. In this case, the Estimator uses the cleaned RSP blocks to get an average estimate with a confidence interval. This estimate in expectation is equal to the true statistic θ' from the unknown entire clean data $\mathbb{D}_{clean}$. Algorithm 1 shows the basic operations to clean block-level samples of dirty RSP blocks and calculate estimates from the cleaned RSP blocks.

Algorithm 1 Estimation from a sample of *dirty* RSP blocks

Input:
- S: block-level sample with g RSP blocks;
- C: data cleaning function;
- f: estimator function;

Operations:
for all $\mathrm{D_q} \in \mathrm{S}$ **do**
  Clean the block: $\mathrm{D'_{\tau_q}} = \mathrm{C(D_{\tau_q})}$;
  Compute a sample estimate: $\hat{\theta}'_q = f(\mathrm{D'_{\tau_q}})$;
  Write $\mathrm{D'_{\tau_q}}$ in HDFS;
end for
Collect the metadata of the new blocks: $\mathrm{S_{clean}} = \{\mathrm{D'_{\tau_q}}\}_{\mathrm{q=1}}^{\mathrm{g}}$
Collect the outputs of f: $\{\hat{\theta}'_q\}_{q=1}^{g}$
Output: $\mathrm{S_{clean}}$, $\{\hat{\theta}'_q\}_{q=1}^{g}$

In addition to the parameters of the target function, e.g., an estimator, a detector, or a cleaner, extra parameters are required including the number of RSP blocks g in a

block-level sample. This can be set according to the number of available cores in the computing cluster to guarantee parallel processing of the selected blocks. In practice, any of these operations can be repeated with multiple samples of RSP blocks in a stepwise process. Furthermore, a data pipeline of these operations can be defined to explore and clean the data.

Theoretical analysis

We introduce a theoretical analysis on using RSP blocks for statistical estimation. First, we define and prove Lemma 1 which states that any record in $\mathbb{D}$ has the same probability to be assigned to any RSP block. Then, we prove that RSP estimates (e.g., mean) are unbiased and consistent estimates in Corollary 1.

Lemma 1 *For any RSP block* $\mathrm{D}_k = \left\{\mathrm{x}_1^{(k)}, \mathrm{x}_2^{(k)}, \ldots, \mathrm{x}_{N_k}^{(k)}\right\}$, $k \in \{1, 2, \ldots, K\}$ *belonging to the RSP* $\mathrm{T} = \{\mathrm{D}_1, \mathrm{D}_2, \ldots, \mathrm{D}_K\}$ *of big data set* $\mathbb{D} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N\}$, $P\{\mathrm{x}_i \in \mathrm{D}_k\} = \frac{N_k}{N}$ *and* $P\left\{\mathrm{x}_j^{(k)} = \mathrm{x}_i\right\} = \frac{1}{N}$ *hold for any* $i \in \{1, 2, \ldots, N\}$ *and* $j \in \{1, 2, \ldots, N_k\}$.

Proof There are $C_N^{N_k}$ possible ways of choosing a subset of N_k records from $\mathbb{D}$. When $\mathrm{x}_i \in \mathrm{D}_k$, $i \in \{1, 2, \ldots, N\}$, there are $C_{N-1}^{N_k-1}$ ways of drawing the remaining $N_k - 1$ records for RSP block D_k. Hence,

$$P\{\mathrm{x}_i \in \mathrm{D}_k\} = \frac{C_{N-1}^{N_k-1}}{C_N^{N_k}} = \frac{N_k}{N}. \tag{1}$$

Furthermore, the N_k data positions of RSP block D_k have the equal possibility to take x_i. Thus,

$$P\left\{\mathrm{x}_j^{(k)} = \mathrm{x}_i\right\} = \frac{1}{N_k} P\{\mathrm{x}_i \in \mathrm{D}_k\} = \frac{1}{N} \tag{2}$$

can be derived for any $j \in \{1, 2, \cdots, N_k\}$. □

Corollary 1 *For any RSP block* $\mathrm{D}_k = \left\{\mathrm{x}_1^{(k)}, \mathrm{x}_2^{(k)}, \ldots, \mathrm{x}_{N_k}^{(k)}\right\}$, $k \in \{1, 2, \ldots, K\}$ *belonging to the RSP* $\mathrm{T} = \{\mathrm{D}_1, \mathrm{D}_2, \ldots, \mathrm{D}_K\}$ *of big data set* $\mathbb{D} = \{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N\}$, *the mean* $\mu^{(k)} = \frac{1}{N_k} \sum_{j=1}^{N_k} \mathrm{x}_j^{(k)}$ *is an unbiased estimator of mean* $\mu = \frac{1}{N} \sum_{i=1}^{N} \mathrm{x}_i$ *of big data set* $\mathbb{D}$.

$$
\begin{aligned}
D\left[\mu^{(k)}\right] &= E\left[\mu^{(k)}-\mu\right]^2 = E\left[\frac{1}{N_k}\sum_{j=1}^{N_k}\mathrm{x}_j^{(k)}-\mu\right]^2 = E\left[\frac{1}{N_k}\sum_{j=1}^{N_k}\left[\mathrm{x}_j^{(k)}-\mu\right]\right]^2 \\
&= \frac{1}{N_k^2}E\left[\sum_{j=1}^{N_k}\left[\mathrm{x}_j^{(k)}-\mu\right]\right]^2 \\
&= \frac{1}{N_k^2}E\left[\sum_{j=1}^{N_k}\left[\mathrm{x}_j^{(k)}-\mu\right]^2+2\sum_{j=1}^{N_k}\sum_{l\neq j}^{N_k}\left[\mathrm{x}_j^{(k)}-\mu\right]\left[\mathrm{x}_l^{(k)}-\mu\right]\right] \\
&= \frac{1}{N_k^2}\left[E\left[\sum_{j=1}^{N_k}\left[\mathrm{x}_j^{(k)}-\mu\right]^2\right]+2E\left[\sum_{j=1}^{N_k}\sum_{l\neq j}^{N_k}\left[\mathrm{x}_j^{(k)}-\mu\right]\left[\mathrm{x}_j^{(k)}-\mu\right]\right]\right] \\
&= \frac{1}{N_k^2}\left[\frac{N_k}{N}\sum_{n=1}^{N}(\mathrm{x}_n-\mu)^2+2\frac{N_k(N_k-1)}{N(N-1)}\sum_{n=1}^{N}\sum_{m\neq n}^{N}(\mathrm{x}_n-\mu)(\mathrm{x}_m-\mu)\right] \\
&= \frac{1}{N_kN}\left[\frac{N-N_k}{N-1}\sum_{n=1}^{N}(\mathrm{x}_n-\mu)^2+\frac{N_k-1}{N-1}\left[\sum_{n=1}^{N}(\mathrm{x}_n-\mu)\right]^2\right] \\
&= \frac{N-N_k}{N_kN(N-1)}\sum_{n=1}^{N}(\mathrm{x}_n-\mu)^2 = \frac{\sum_{n=1}^{N}(\mathrm{x}_n-\mu)^2}{N-1}\times\frac{1}{N_k}\times\frac{N-N_k}{N}
\end{aligned}
\tag{4}
$$

Proof From Lemma 1, we can get $P\left\{\mathrm{x}_j^{(k)}=\mathrm{x}_i\right\}=\frac{1}{N}$ for any $i\in\{1,2,\ldots,N\}$ and $j\in\{1,2,\ldots,N_k\}$. Hence, we can derive:

$$
\begin{aligned}
E\left[\mu^{(k)}\right] &= E\left[\frac{1}{N_k}\sum_{j=1}^{N_k}\mathrm{x}_j^{(k)}\right] = \frac{1}{N_k}\sum_{j=1}^{N_k}E\left[\mathrm{x}_j^{(k)}\right] \\
&= \frac{1}{N_k}\sum_{j=1}^{N_k}\left[\frac{1}{N}\sum_{i=1}^{N}\mathrm{x}_i\right] = \frac{1}{N}\frac{1}{N_k}\sum_{j=1}^{N_k}\sum_{i=1}^{N}\mathrm{x}_i \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathrm{x}_i = \mu.
\end{aligned}
\tag{3}
$$

It indicates that $\mu^{(k)}$ is an unbiased estimator of mean μ, i.e., the estimator $\mu^{(k)}$ has the unbiasedness. □

Based on Corollary 1, we can further derive that $\mu^{(k)}$ is a consistent estimator of mean μ, i.e., the estimator $\mu^{(k)}$ has the consistency. The variance of mean $\mu^{(k)}$ can be expressed as Eq. (4). According to Chebyshev's inequality, we can get

$$
P\left\{\left|\mu^{(k)}-\mu\right|\geq\varepsilon\right\}\leq\frac{D\left[\mu^{(k)}\right]}{\varepsilon^2} \tag{5}
$$

Table 1 Datasets

Name	*N*	*M*	*K*	*n*
HIGGS	11,000,000	28	100	110,000
Power	46,669,266	99	6667	7000
AirOnTime87to12	148,619,655	47	298	498,724

Characteristics of the real data sets and the RSPs generated for them

holds for any given $\varepsilon > 0$. Furthermore, the inequality can be transformed to

$$P\left\{\left|\mu^{(k)} - \mu\right| < \varepsilon\right\} \geq 1 - \frac{D\left[\mu^{(k)}\right]}{\varepsilon^2}. \tag{6}$$

When the mean and variance of the population exist, the variance $\sigma^2 = \frac{\sum_{i=1}^{N}(\mathrm{x}_i-\mu)^2}{N-1}$ of $\mathbb{D}$ is an unbiased estimator of the population variance. It is reasonable to assume that σ^2 is bounded. Hence, we can derive

$$P\left\{\left|\mu^{(k)} - \mu\right| < \varepsilon\right\} \geq 1 - \frac{\sigma^2}{N_k\varepsilon^2} \tag{7}$$

and

$$\lim_{N_k \to +\infty} P\left\{\left|\mu^{(k)} - \mu\right| < \varepsilon\right\} = 1. \tag{8}$$

In conclusion, we prove that the mean of an RSP block D_k is an unbiased and consistent estimator of the mean of $\mathbb{D}$. As a result, RSP blocks can be used to get an approximate sampling distribution of a sample statistic and get an average estimate within a confidence interval.

Results

In this section, we demonstrate the performance of RSP-Explore in univariate (location, variability/dispersion, skew, kurtosis) and bivariate exploration (correlation) from both clean and dirty data.

Experiment environment and settings

We tested RSP-Explore on a small computing cluster of 5 nodes with Apache Spark 1.6 and Microsoft R Server 9.1. We used our Spark implementation of the two-stage data partitioning algorithm [42] to generate an HDFS-RSP file from each data set. The characteristics of the three data sets and the RSPs generated for this experiment are listed in Table 1. For statistical estimation of summary statistics and correlation coefficients, we applied existing sequential R functions to a sample of RSP blocks in parallel. Each sample of g RSP blocks was executed as a single Spark job with only one stage of g Spark tasks. On the other hand, we applied the parallelized functions from RevoScaleR[7] package to the entire data to get the true statistics and compare with the results from block-level samples of RSP blocks. To quantify the uncertainty of the RSP-based estimates, we used percentiles to calculate the confidence interval of an RSP-based estimate. In this paper,

[7] https://docs.microsoft.com/en-us/machine-learning-server/r-reference/revoscaler/revoscaler.

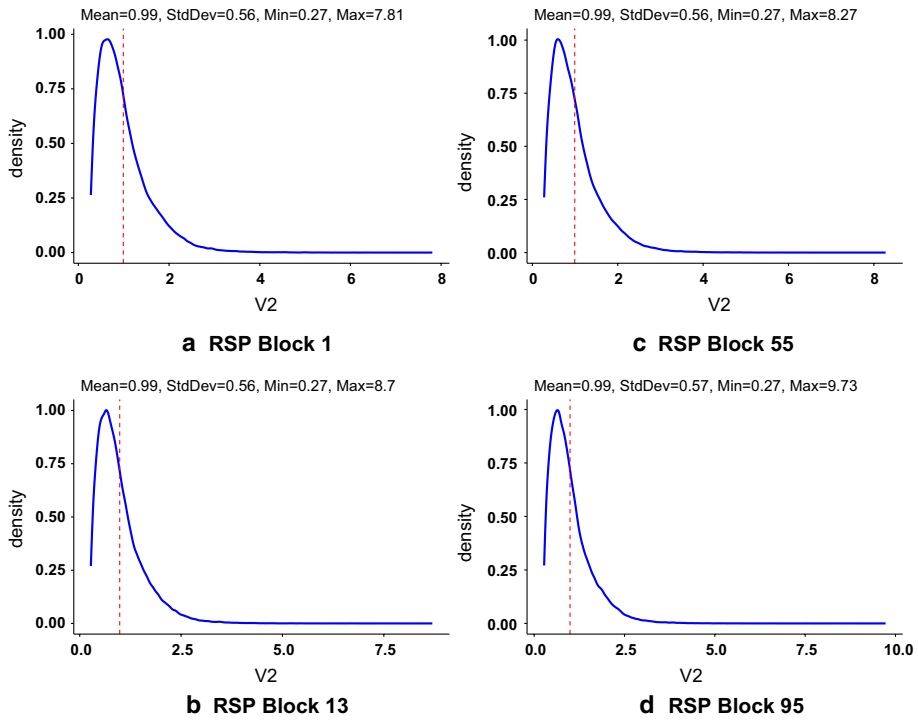

Fig. 4 Exploring the data distribution in HIGGS. Density plots of V2 in 4 RSP blocks from HIGGS data

we report the results within 90% confidence level, i.e., between the 5th and 95th percentiles of the sampling distribution. We also show error bars from multiple runs of the stepwise process to show the variance of the results. For error detection and data cleaning, we implemented the error detector and data cleaner functions as sequential R functions and applied them to a sample of RSP blocks in a similar way to existing R functions. On the other hand, we used the rxDataStep transformation in RevoScaleR to apply validation and cleaning rules to the entire data and get the true proportions and the entire clean data. For this experiment, we configured the cluster to use 96 cores. Thus, to avoid the latency in processing RSP blocks, the number of RSP blocks g in a block-level sample should not largely exceed 96. Since HIGGS data has only 100 RSP blocks, we tested the RSP-Explore using block-level samples with only $g = 5$ RSP blocks to show how the results change after each sample. For Power data, we used block-level samples with $g = 100$ RSP blocks.

Exploring HIGGS data

For this experiment, we generated an RSP from HIGGS[8] data with $K = 100$ RSP blocks and $n = 110{,}000$ records in each block. We started the experiment by exploring the data distribution and summary statistics in some RSP blocks. These blocks have distributions as shown in Fig. 4 and there is no significant difference between the summary statistics values from these blocks. We used the Statistics Estimator and the Error Detector to further explore this data. As an example, we consider feature V2 that represents *lepton pT* in HIGGS data. We summarize the results as follows:

[8] https://archive.ics.uci.edu/ml/datasets/HIGGS.

Table 2 Summary statistics of V2 in HIGGS data

Stat	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Mean	0.9914658	0.9916447 ± 0.001691498	0.9892427	0.9921890	0.9941837
StdDev	0.5653777	0.5657651 ± 0.002567618	0.5625647	0.5652471	0.5696894
Median	0.8533714	0.8531884 ± 0.001508743	0.8513674	0.8530054	0.8555858
MAD	0.4485073	0.4483536 ± 0.001575363	0.4464044	0.4482361	0.4509222
Skewness	1.758388	1.758561 ± 0.03039423	1.719768	1.753431	1.810040
Kurtosis	5.57178304	5.551798 ± 0.4035557	5.048026	5.532661	6.335199

RSP-based estimates were calculated from a sample of RSP blocks ($g = 15$)

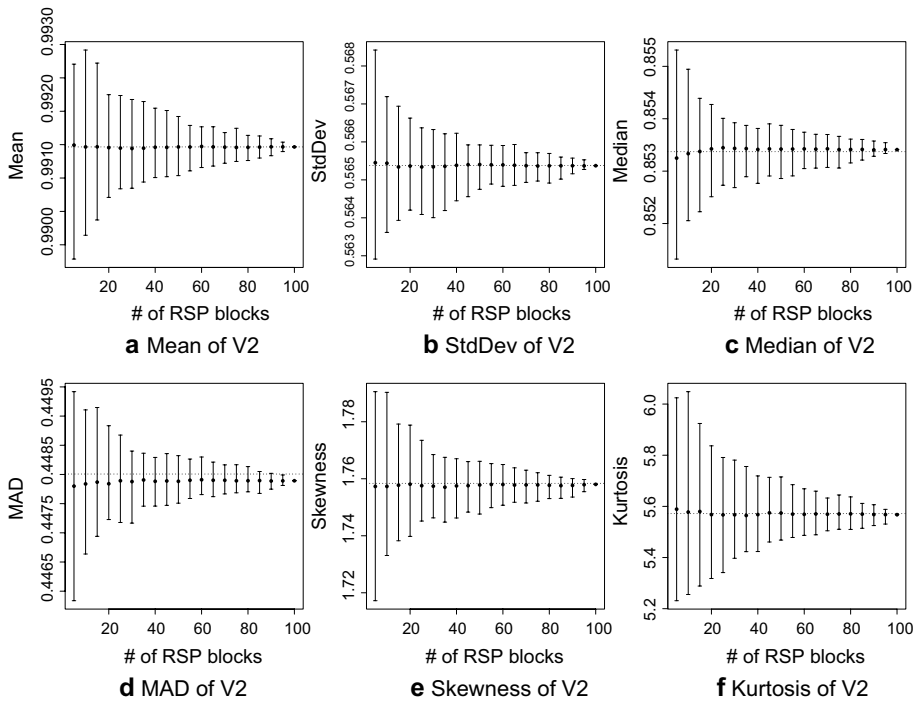

Fig. 5 Summary statistics of V2 in HIGGS. Incremental estimation of summary statistics of V2 using samples of RSP blocks of HIGGS data ($g = 5$). The dotted line represents the true value of the statistic calculated from the entire HIGGS data

Summary statistics

In Table 2, we compare the true summary statistics (mean, standard deviation, median, MAD, skewness, and kurtosis) of feature V2 with the RSP-based estimates from $g = 15$ RSP blocks. We can observe that there is no significant difference between the RSP-based estimates and the true values. The estimates have small variance and narrow confidence interval. To show how the RSP-based estimates change when adding more RSP blocks, we ran the Estimator incrementally on block-level samples with 5 RSP blocks until all RSP blocks were used up. We notice that the estimated value converges quickly to the true value and the error range becomes narrower with more RSP blocks as shown in Fig. 5.

Correlation

Table 3 shows the correlation coefficients between V2 and 6 other features in HIGGS data. We also compare the estimated coefficients from $g = 15$ RSP blocks with the true values. Figure 6 shows how the estimated coefficients change with more RSP blocks. Similarly to summary statistics, the same observation applies in case of correlation coefficients.

Error detection

We used the Detector to check whether HIGGS data has inconsistent values. For instance, Table 4 shows that V2 has no errors and no missing values, but there is a small proportion of outliers. In this example, negative values are errors and outliers are within $3 * MAD$ of the median. Since HIGGS is not very big and there is only a small proportion

Table 3 Correlation between V2 and 6 other features in HIGGS data

Feature	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
V3	− 0.000153	− 0.000151 ± 0.003051	− 0.003929	− 0.000420	0.002098
V10	− 0.006265	− 0.006065 ± 0.003414	− 0.011673	− 0.006966	− 0.002738
V15	− 0.011190	− 0.011556 ± 0.003186	− 0.015670	− 0.011981	− 0.006128
V20	0.000090	0.000023 ± 0.003806	− 0.003273	0.000171	0.007839
V25	0.272327	0.274107 ± 0.003175	0.266316	0.270652	0.277722
V29	0.141168	0.141790 ± 0.003782	0.135513	0.141922	0.146332

RSP-based coefficients were calculated from a sample of RSP blocks ($g = 15$)

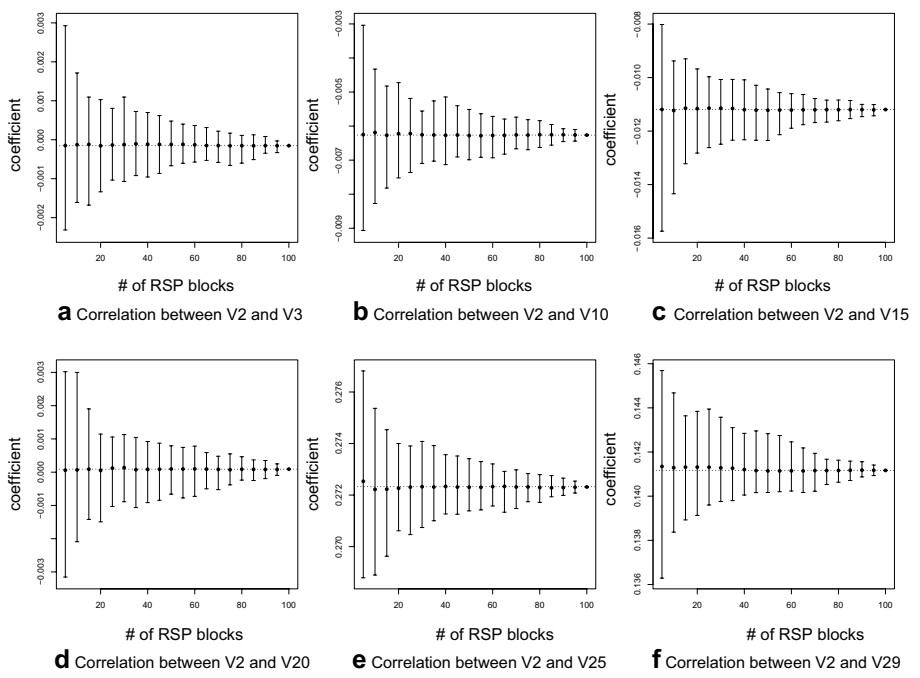

Fig. 6 Correlation between V2 and 6 other features in HIGGS. Incremental estimation of Pearson's correlation coefficients between V2 and 6 other features using samples of RSP blocks of HIGGS data ($g = 5$). The dotted line in each plot represents the coefficient computed from the entire HIGGS data

Table 4 Proportions of errors, outliers, missing and valid values in V2 in HIGGS data

Category	True value	Summary of RSP-based proportions			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Errors	0	0 ± 0	0	0	0
Outliers	0.04084664	0.04082 ± 0.0006663723	0.03977546	0.04074545	0.04160728
Missing values	0	0 ± 0	0	0	0
Valid values	0.95915336	0.95918 ± 0.0006663723	0.9583927	0.9592546	0.9602245

RSP-based proportions were calculated from a sample of RSP blocks ($g = 15$)

of outliers, we don't show the effect of data cleaning in this case. In the following section, we demonstrate this point with Power data.

Exploring Power data

Power data was extracted from a smart grid database of an industrial area in Guangdong province. This data contains 46,669,266 records with 98 features: smart meter identifier, date, and the remaining 96 features represent power consumption every 15 minutes in a day. For this experiment, we created an RSP with $K = 6667$ RSP blocks and nearly $n = 7000$ records in each RSP block.

We started the experiment by exploring the data distribution and summary statistics in some RSP blocks. Figure 7 shows the density plots of V38 in 4 RSP blocks. In these plots, we limit the maximum value of V38 to 50, 000 to show the distribution clearly. We can see that there is no significant difference between the mean values from different RSP blocks. However, the standard deviation values are not consistent. It is so high when there are extreme values such as those in blocks 1528 and 2569 in the Figure. To

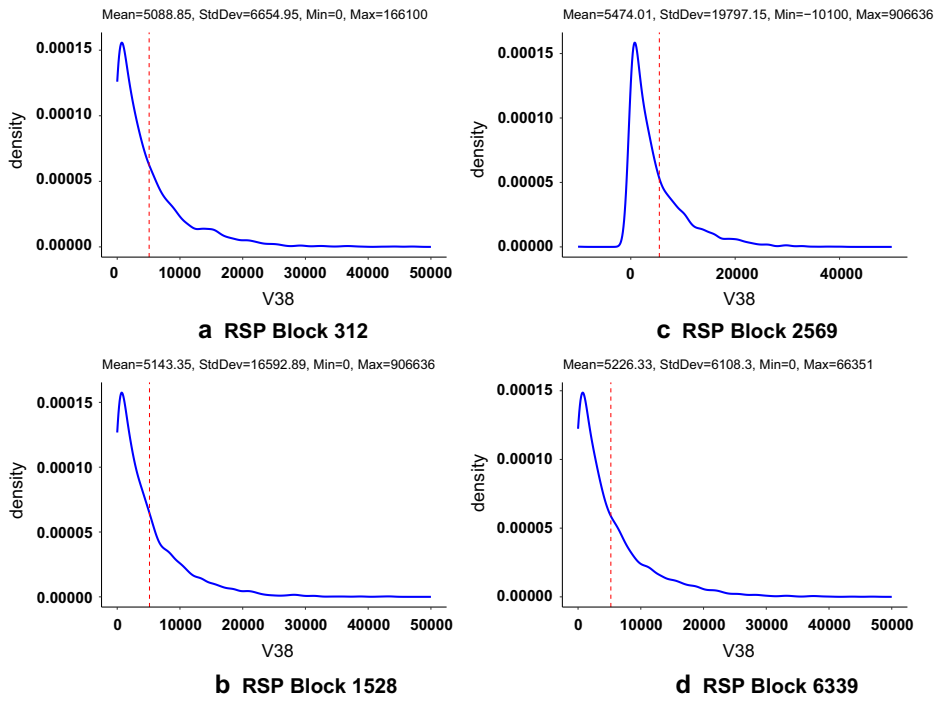

Fig. 7 Exploring the data distribution in Power data. Density plots of V38 in 4 RSP blocks from Power data

investigate more about this data, we used block-level samples with $g = 100$ RSP blocks in the estimator, detector and cleaner. In this paper, we show only the results for only one feature, V38. We summarize our findings as follows:

Summary statistics

Table 5 compares the RSP-based statistics of V38 from 100 RSP blocks and the true values from the entire data. While the estimated median and MAD are close to the true values and with narrow confidence intervals, other statistics differ significantly from the true values and with large confidence intervals. This is expected especially with the high standard deviation of V38 in some RSP blocks due to extreme values. To show how the RSP-based estimates from this dirty Power data change when adding more RSP blocks, we ran the Estimator incrementally on block-level samples with 100 RSP blocks until all RSP blocks were used up. Figure 8 shows the results from 25 runs of this process for

Table 5 Summary statistics of feature V38 in Power data

Stat	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Mean	5314.49	5256.008 ± 215.3196	4942.404	5254.394	5602.095
StdDev	15127.22	10937.52 ± 4318.59	5990.154	10331.031	19210.659
Median	3100	3103.9 ± 88.62217	2982.475	3099.750	3243.750
MAD	3783.595	3776.738 ± 106.9769	3616.877	3783.966	3954.020
Skewness	− 108.5203	20.84063 ± 12.67476	2.474514	21.064943	39.870585
Kurtosis	71059.13	802.1309 ± 660.3176	11.966831	692.604567	1977.642239

RSP-based estimates were calculated from a sample of RSP blocks ($g = 100$)

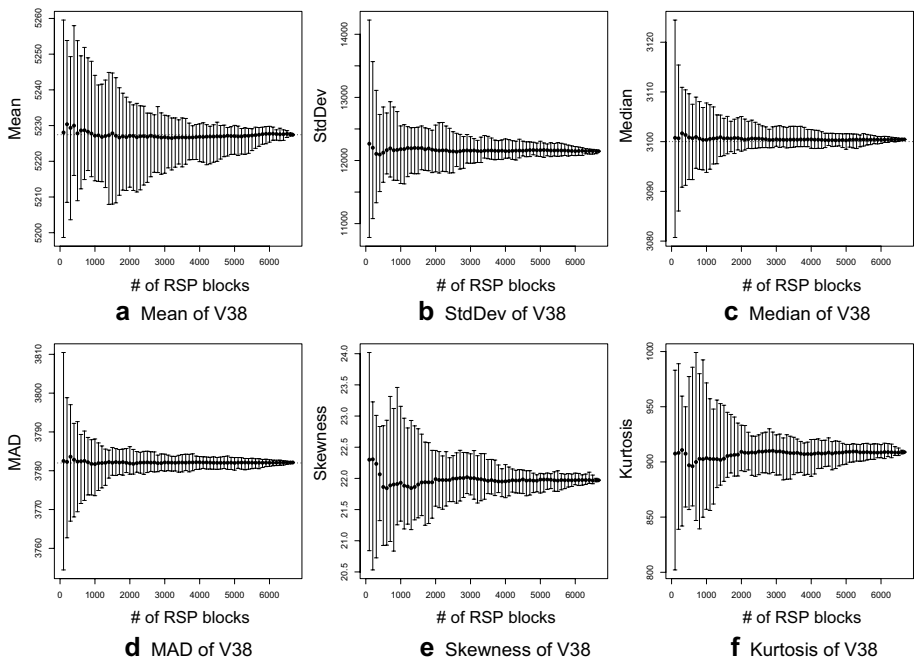

Fig. 8 Summary statistics of V38 in power. Incremental estimation of summary statistics of V38 using samples of RSP blocks of Power data ($g = 100$)

each statistic. The estimates of the mean, median and MAD converge quickly to the true values. In case of the standard deviation, skewness, and kurtosis, the estimate converges to a value that differs significantly from the true value.

Correlation

As the features in this data represent power consumption at certain time points in a day, we can expect that each feature has higher correlation with nearby features. We used the Estimator to explore the correlation coefficients between features. For instance, Table 6 shows the RSP-based coefficients between V38 and 6 other features. To show how the correlation coefficients change with more data, we ran the Estimator incrementally to compute the correlation coefficients from block-level samples with 100 RSP blocks until all the blocks were used up. The plots in Fig. 9 shows the results for the correlation between V38 and 6 other features. In a similar way to the RSP-based summary statistics,

Table 6 Correlation between V38 and 6 other features in Power data

Feature	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
V3	0.8512235	0.7536487 ± 0.1524914	0.5233034	0.7376381	0.9631805
V15	0.8720032	0.8009430 ± 0.1265738	0.6350162	0.7963755	0.9701548
V37	0.96786065	0.97107535 ± 0.02932448	0.93525808	0.97572809	0.99619922
V39	0.95651256	0.96303372 ± 0.02644475	0.92738897	0.96685111	0.99514627
V65	0.8905704	0.8496362 ± 0.0984346	0.6913239	0.8548659	0.9741665
V98	0.8527415	0.7653636 ± 0.1444491	0.5650703	0.7657670	0.9650463

RSP-based coefficients were calculated from a sample of RSP blocks ($g = 100$)

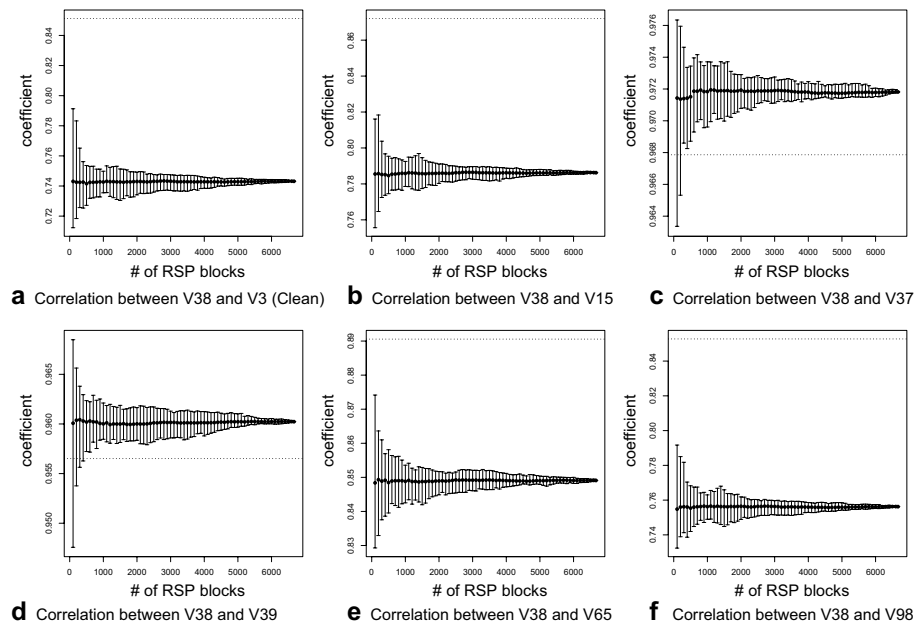

Fig. 9 Correlation between V38 and 6 other features in power. Incremental estimation of Pearson's correlation coefficients between V38 and 6 other features using samples of RSP blocks of Power data ($g = 100$). The dotted line in each plot represents the coefficient computed from the entire data

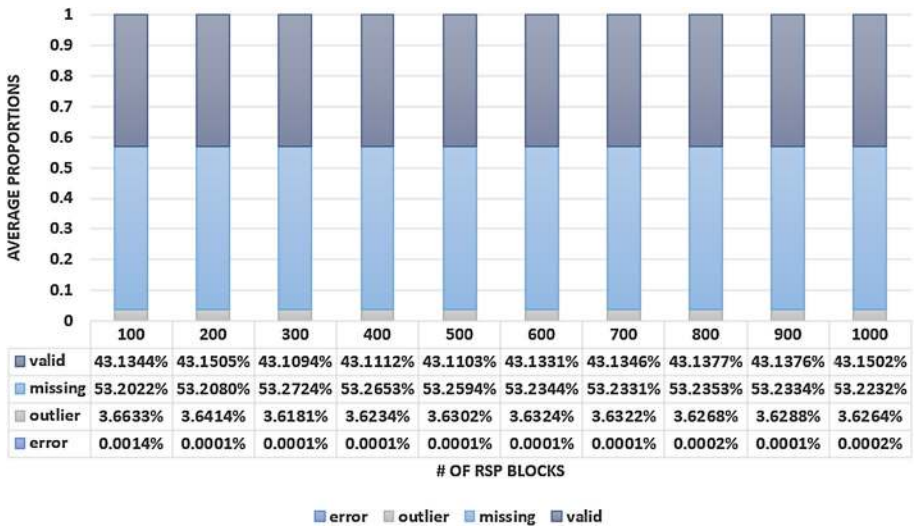

Fig. 10 Proportions of V38. Average proportions of errors, outliers, missing, and valid values of V38 from incremental batches of RSP blocks (from 100 to 1000)

Table 7 Proportions of errors, outliers, missing and valid values in V38

Category	True value	Summary of RSP-based proportions			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Errors	0.00000154	0.00001408 ± 0.00001408	0	0	0
Outliers	0.03630323	0.03663272 ± 0.002324408	0.03328592	0.03656265	0.04053885
Missing values	0.53230704	0.5320224 ± 0.005352080	0.52315101	0.53212264	0.54086298
Valid values	0.43138819	0.4313435 ± 0.005839722	0.42173133	0.43105753	0.44055828

RSP-based proportions were calculated from a sample of RSP blocks ($g = 100$)

we notice that the estimated correlation coefficient converge quickly and adding more data doesn't change the value significantly. However, RSP-based coefficients are not so close to the values from the entire data. Since Pearson's correlation coefficient depends on the standard deviation, it is sensitive to outliers.

Error detection

Summary statistics from a block-level sample of RSP blocks revealed that Power data has inconsistent values. To investigate more about these inconsistent values, we defined three validation rules on each of the power consumption features: (1) negative values are considered as errors, (2) outliers are values that are more than $3 * MAD$ from the median, and (3) missing values are represented as *NAs*. We estimated the average proportions from block-level samples with $g = 100$ RSP blocks and updated the results incrementally. Figure 10 shows that the estimated proportions don't vary significantly with more RSP blocks. In fact, the average proportions from 100 RSP blocks are comparable to those computed from the entire data as shown in Table 7. Similarly to the summary statistics, incremental proportions with error ranges are shown in Fig. 11.

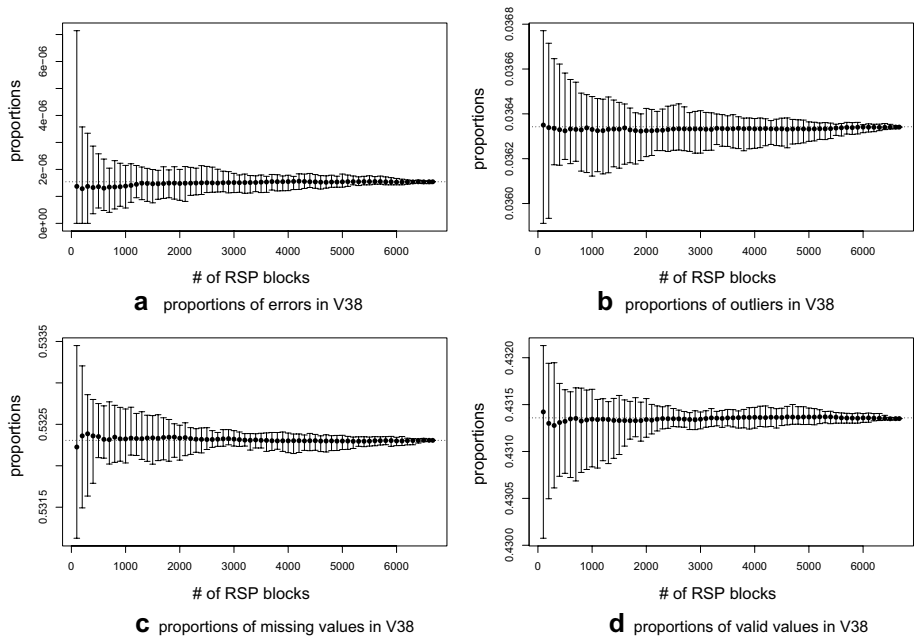

Fig. 11 Proportions of V38. Incremental estimation of proportions in V38 using samples of RSP blocks of Power data ($g = 100$)

Data cleaning

We applied the same cleaning rules to the three categories: errors, outliers and missing values. Since the proportion of these values is high, we replaced them with the median value. We used the Data Cleaner to apply these rules to only the first block-level sample of $g = 100$ RSP blocks and recomputed the summary statistics and correlation coefficients from the cleaned blocks. For comparison, we also applied the same rules to the entire Power data and recomputed the true values. Table 8 shows the summary statistics of V38 from the cleaned RSP blocks and the true statistics from the entire clean data. Similarly, Table 9 shows the correlation coefficients between V38 and the 6 other features in the cleaned data. To show how the estimated values change with more data, we ran Algorithm 1 incrementally using block-level samples with $g = 100$ RSP blocks. Figure 12 shows the results from 25 runs of this process for each statistic and Fig. 13 shows the results for the correlation between V38 and the 6 other features. The estimated values from clean data have the expected characteristics of RSP-based estimates, e.g., converge quickly with small error bounds. This shows the effect of data cleaning and that only a few clean RSP blocks are sufficient to explore the statistical properties of the entire unknown clean data.

Exploring airlines data

For this experiment, we generated an RSP from the Airlines performance data, AirOnTime87to12[9], with $K = 298$ RSP blocks and $n = 500,000$ records in each block. Then, we used RSP-Explore to explore different features about flights delay such as ArrDelay.

[9] https://packages.revolutionanalytics.com/datasets/AirOnTime87to12/.

Table 8 Summary statistics of feature V38 in *clean* Power data

Stat	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Mean	3404.014	3396.572 ± 101.0598	3241.725	3403.829	3559.236
StdDev	2388.191	2382.172 ± 69.43742	2271.915	2378.807	2486.027
Median	3100	3103.9 ± 88.62217	2982.475	3099.750	3243.750
MAD	0	0 ± 0	0	0	0
Skewness	1.996612	1.996031 ± 0.03404484	1.947026	1.997957	2.053139
Kurtosis	5.185274	5.187026 ± 0.2038164	4.842079	5.144150	5.537230

RSP-based estimates were calculated from a sample of *clean* RSP blocks ($g = 100$)

Table 9 Correlation between V38 and 6 other features in *clean* Power data

Feature	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
V3	0.3126873	0.3132926 ± 0.01267681	0.2956115	0.3131225	0.3356508
V15	0.3268368	0.3276999 ± 0.01238780	0.3059839	0.3286929	0.3459587
V37	0.8598520	0.8601381 ± 0.01011005	0.8427635	0.8606978	0.8760539
V39	0.5330211	0.5335060 ± 0.01609916	0.5082045	0.5346581	0.5578231
V65	0.63534	0.6340810 ± 0.01368082	0.6108725	0.6361881	0.6562849
V98	0.4790145	0.4784269 ± 0.01759861	0.4507184	0.4788393	0.5072602

RSP-based coefficients were calculated from a sample of *clean* RSP blocks ($g = 100$)

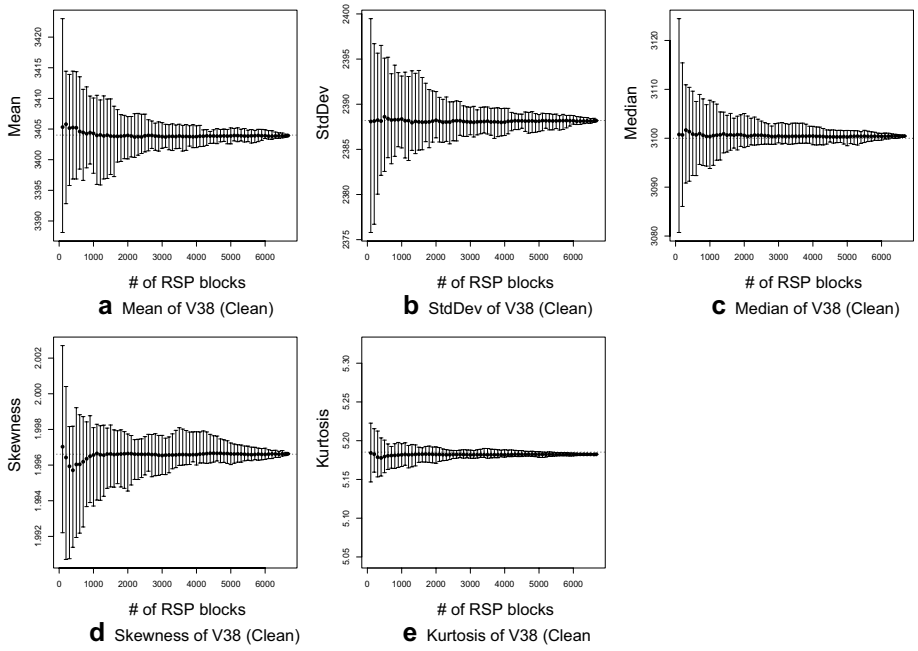

Fig. 12 Summary statistics of V38 in clean Power data. Incremental estimation of summary statistics of V38 using samples of *clean* RSP blocks of Power data ($g = 100$)

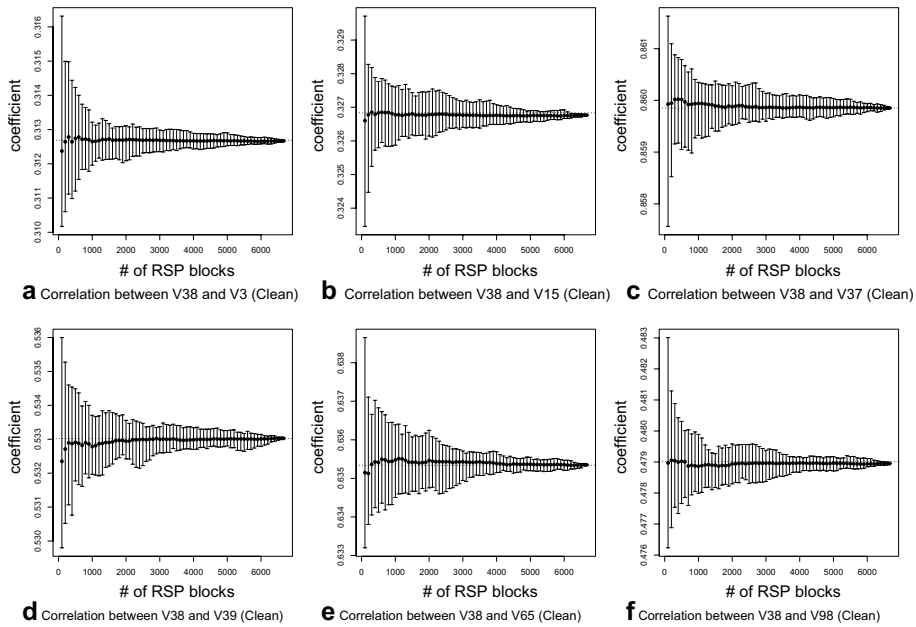

Fig. 13 Correlation between V38 and 6 other features in clean Power data. Incremental estimation of Pearson's correlation coefficients between V38 and 6 other features using samples of *clean* RSP blocks of Power data ($g = 100$)

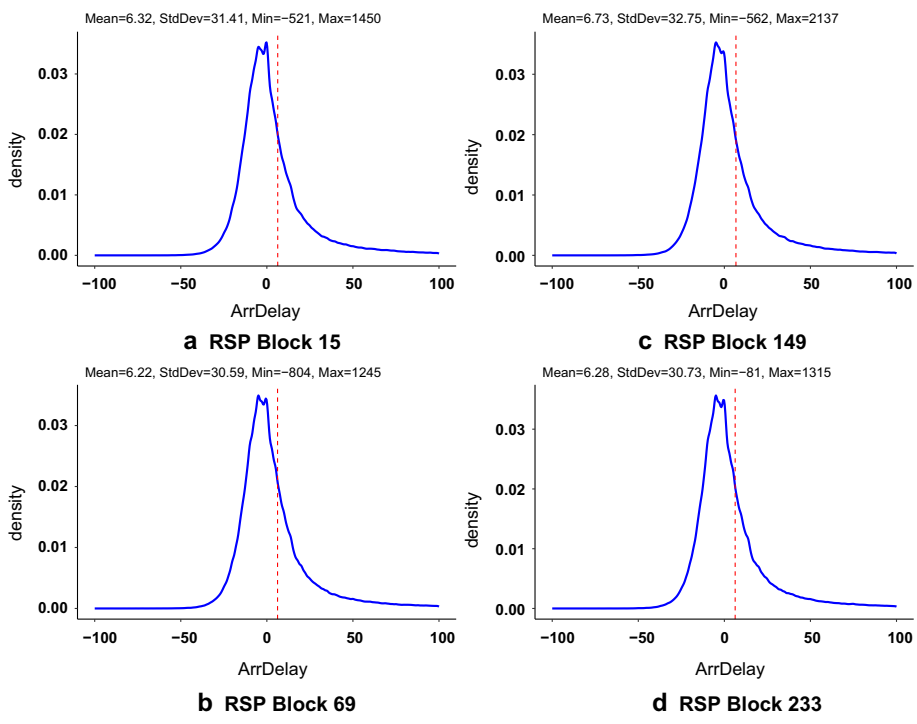

Fig. 14 Exploring the data distribution in AirOnTime87to12. Density plots of ArrDelay in 4 RSP blocks from AirOnTime87to12 data

Figure 14 shows the density plots of the ArrDelay feature in four RSP blocks. We notice the similar distributions in these blocks. We used a block-level sample with only $g = 10$ RSP blocks to estimate the statistical properties of the entire data. Table 10 compares true statistics with RSP-based estimates. Similarly, the true correlation coefficients between ArrDelay and 6 other features in the data are compared with the RSP-based coefficients in Table 11. We also used the Error Detector to estimate the proportions of inconsistent values in ArrDelay. As shown in Table 12, this feature has a small proportion of outliers and missing values. From the previous tables we can see that 10 RSP blocks of the AirOnTime87to12 data can be used effectively to get summary statistics and proportions with narrow confidence intervals. These approximate results help data scientists decide on how the data should be cleaned in a similar way to Power data.

Table 10 Summary statistics of ArrDelay in AirOnTime87to12 data

Stat	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Mean	6.5666	6.471984 ± 0.1872946	6.229485	6.518680	6.728294
StdDev	31.55641	31.58147 ± 0.5532607	30.73029	31.59074	32.25136
Median	0	0 ± 0	0	0	0
MAD	13.3434	13.3434 ± 0	13.3434	13.3434	13.3434
Skewness	5.763419	5.685329 ± 0.3536372	5.189962	5.765824	6.116367
Kurtosis	90.08612	88.75798 ± 17.53257	68.61477	87.41795	115.98568

RSP-based estimates were calculated from a sample of RSP blocks ($g = 10$)

Table 11 Correlation between ArrDelay and 6 other features in Airlines data

Feature	True value	Summary of RSP-based estimates			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
DepDelay	0.9257360	0.9266382 ± 0.0098338	0.906661909	0.929922510	0.936090184
WeatherDelay	0.2616046	0.2592562 ± 0.0233771	0.228037031	0.258204030	0.297533928
Distance	− 0.0134785	− 0.0151175 ± 0.0123154	− 0.034275887	− 0.016134390	0.006646666
AirTime	− 0.0146634	− 0.0170937 ± 0.0124837	− 0.036912348	− 0.019335688	0.004608020
CarrierDelay	0.5357752	0.5348132 ± 0.0297866	0.489324517	0.534856336	0.586338471
NASDelay	0.3217719	0.3181037 ± 0.0243393	0.284709646	0.320763065	0.352731115

RSP-based coefficients were calculated from a sample of RSP blocks ($g = 10$)

Table 12 Proportions of errors, outliers, missing and valid values in ArrDelay in AirOnTime87to12 data

Category	True value	Summary of RSP-based proportions			
		Mean ± StdDev	5th percentile	50th percentile	95th percentile
Errors	0	0 ± 0	0	0	0
Outliers	0.07903821	0.07931381 ± 0.00185624	0.07680381	0.07920513	0.08183604
Missing values	0.02047453	0.02039184 ± 0.00111821	0.01903729	0.02016446	0.02201047
Valid values	0.90048725	0.9002944 ± 0.00227722	0.8977497	0.9003487	0.9037174

RSP-based proportions were calculated from a sample of RSP blocks ($g = 10$)

Discussion

In the previous experiments, we used three real data sets to demonstrate the statistical advantages of the RSP-Explore method. We can see that a few RSP blocks are sufficient to explore both clean and dirty data (less than 15% of HIGGS, 1.5% of Power, and 3.5% of AirOnTime87to12). In average, the processing time is reduced from minutes to seconds for these data sets on our small computing cluster with 5 nodes. Since data scientists iteratively apply a variety of techniques to explore a data set, the RSP-Explore method can lead to significant implications on the scalability and efficiency of big data exploration tasks and the productivity of data scientists. This is basically due to the underlying RSP data model that facilitates online random sampling from big data in cluster computing frameworks [37].

In this paper, we demonstrate the performance of RSP-Explore on numerical data. However, the same method can be used with categorical data. For instance, a block-level sample of RSP blocks can be used to estimate the relative frequencies of the categories in a categorical feature. Similarly, the proportion of erroneous categorical values can be estimated in the same way as we estimate the proportions of inconsistent values in numerical data. Furthermore, RSP-Explore can be extended directly to support logical data cleaning tasks such as those discussed in [15, 54]. A block-level sample can be used to estimate the proportion of records that don't satisfy a certain constraint or the proportion of values that are slightly different from the correct value. In this case, the estimated proportions can help data scientists decide on the repairing strategy or whether to tolerate small differences in the values. Since RSP-Explore can work on small batches of RSP blocks, it can also be employed for interactive data cleaning where data scientists correct the violations of integrity constraints in a block-level sample of RSP blocks. This is necessary for human-in-the-loop data analysis [55–58].

In principle, this method can be applied to any multivariate data set where objects are represented by one or more features and stored in a tabular format. This is a common form for representing different types of data whether the source data is structured, semi structured or unstructured. For instance, a corpus of text data is usually represented in a document-term matrix that is similar to a data frame with records representing the documents and features representing the terms. This matrix can be stored as an RSP. Then, the RSP-Explore method can be applied, for instance, to estimate the distribution of a term frequency in the entire corpus using only a block-level sample of RSP blocks. This can be used to identify potential stop words in the corpus.

As we demonstrated in this paper, RSP-Explore is a good method to quickly understand the global statistical properties in a big data set using existing sequential or user-define functions. The basic statistics and proportions are estimated from a block-level sample without computing the entire data. The number of RSP blocks g in a block-level sample can be determined according to the available cores in the cluster. Unfortunately, these computational and statistical advantages can't be obtained directly in some cases:

- Currently, RSP-Explore can't be used to get an approximate histogram. While it is possible to get histograms of individual RSP blocks, building an approximate histogram requires criteria for combining local histograms and quantifying the uncertainty of the approximated global histogram. We are currently working on extend-

ing RSP-Explore to build an approximate equi-width histogram that can be used to quickly understand the probability distribution of the entire data;

- RSP-Explore can't be used directly to detect and repair duplication errors. It needs an additional step to check duplications across RSP blocks. We are currently experimenting this idea. Furthermore, empirical and theoretical evidences are necessary to study the affect of de-duplication on the probability distribution in RSP blocks and the similarity between these blocks and the entire unknown clean data.In fact, big data cleaning burden would dramatically be alleviated if repairing duplicates in only a small block-level sample was enough to get samples of the entire unknown clean data.
- RSP-Explore is not designed for streaming data. As we mentioned before, we target at offline workloads where data scientists explore big data sets on computing clusters with a variety of techniques. For steaming data, a different strategy is required to get synopses of the data such as sketching [59].
- If the target is to find statistics or proportions in a certain subspace in the data, we may need alternative data partitioning algorithms to create RSP blocks with specific characteristics (e.g., each block is a random sample of the observations about customers in a certain city or branch). This issue still needs more investigation and experiments.

Conclusions

In this paper, we presented the RSP-Explore method for big data exploration and cleaning on small computing clusters. This method addresses three main tasks using the RSP approach: statistical estimation, error detection and data cleaning. With this method, data scientists can tune the amount of processed data according to the available cores in a computing cluster. We demonstrated that a few RSP blocks are enough to explore the statistical properties of a big data set including summary statistics and proportions of inconsistent values. The experimental results of three real data sets show that RSP-based estimates can rapidly converge toward the true values and that cleaning a sample of RSP blocks is enough to estimate the statistical properties of the unknown clean data. Some of our current and future works include using the RSP approach for histogram estimation, data visualization, and feature selection. In addition, we are experimenting alternative data partitioning algorithms to generate RSP blocks on small computing cluster.

Abbreviations

RSP: random sample partition; HDFS: Hadoop Distributed File System; RLS: record-level sampling; BLS: block-level sampling; SDF: sample distribution function.

Acknowlegements

We sincerely thank the editors and anonymous reviewers whose valuable comments and suggestions helped us improve this paper significantly.

Authors' contributions

SS performed the primary work and experiments of this manuscript. JZH proposed the main idea, took on a supervisory role and oversaw the completion of the work. YH did the theoretical analysis. All authors read and approved the final manuscript.

Funding
This paper was supported by the National Natural Science Foundation of China (61473194), National Key R&D Program of China (2017YFC0822604-2), China Postdoctoral Science Foundation (2016T90799) and Scientific Research Foundation of Shenzhen University for Newly-introduced Teachers (2018060).

Availability of data and materials
The HIGGS dataset analysed during the current study is available in the UCI Machine Learning repository, https://archive.ics.uci.edu/ml/datasets/HIGGS. The AirOnTime87to12 dataset analysed during the current study is available online at https://packages.revolutionanalytics.com/datasets/AirOnTime87to12/. The Power dataset analysed during the current study is not publicly available.

Competing interests
The authors declare that they have no competing interests.

Author details
[1] Big Data Institute, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. [2] National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, China.

Received: 21 February 2019 Accepted: 10 May 2019
Published online: 04 June 2019

References

1. Nair R. Big data needs approximate computing: technical perspective. Commun ACM. 2014;58(1):104. https://doi.org/10.1145/2688072.
2. Goiri Í, Bianchini R, Nagarakatte S, Nguyen TD. Approxhadoop: bringing approximations to mapreduce frameworks. In: Proceedings of the twentieth international conference on architectural support for programming languages and operating systems. ACM; 2015. p. 383–97.
3. Salloum S, Huang JZ, He Y. Empirical analysis of asymptotic ensemble learning for big data. In: Proceedings of the 3rd IEEE/ACM international conference on big data computing, applications and technologies. BDCAT '16, ACM, New York, NY, USA; 2016, p. 8–17. https://doi.org/10.1145/3006299.3006306.
4. Kwon BC, Verma J, Haas PJ, Demiralp. Sampling for scalable visual analytics. IEEE Comput Graph Appl. 2017;37(1):100–8. https://doi.org/10.1109/MCG.2017.6.
5. Riondato M. Sampling-based data mining algorithms: modern techniques and case studies. In: Proceedings of the 2014th European conference on machine learning and knowledge discovery in databases-volume part III. ECMLPKDD'14, Berlin, Heidelberg: Springer; 2014, p. 516–9.
6. Krishnan S, Wang J, Franklin MJ, Goldberg K, Kraska T, Milo T, Wu E. Sampleclean: fast and reliable analytics on dirty data. IEEE Data Eng Bull. 2015;38(3):59–75.
7. Sutton CA, Hobson T, Geddes J, Caruana R. Data diff: interpretable, executable summaries of changes in distributions for data wrangling. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, KDD 2018, London, UK, August 19–23, 2018; 2018. p. 2279–88. https://doi.org/10.1145/3219819.3220057.
8. Chu X, Ilyas IF, Krishnan S, Wang J. Data cleaning: overview and emerging challenges. In: Proceedings of the 2016 international conference on management of data. SIGMOD '16. ACM, New York, NY, USA; 2016, p. 2201–6. https://doi.org/10.1145/2882903.2912574.
9. Krishnan S, Wang J, Wu E, Franklin MJ, Goldberg K. Activeclean: interactive data cleaning for statistical modeling. Proc VLDB Endow. 2016;9(12):948–59. https://doi.org/10.14778/2994509.2994514.
10. Tukey JW. Exploratory data analysis. Behavioral science: quantitative methods. Addison-Wesley, Reading: Mass; 1977.
11. Rojas JAR, Kery MB, Rosenthal S, Dey A. Sampling techniques to improve big data exploration. In: 2017 IEEE 7th symposium on large data analysis and visualization (LDAV); 2017, p. 26–35. https://doi.org/10.1109/LDAV.2017.8231848.
12. Idreos S, Papaemmanouil O, Chaudhuri S. Overview of data exploration techniques. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data. SIGMOD '15. ACM, New York, NY, USA; 2015, p. 277–81. https://doi.org/10.1145/2723372.2731084.
13. Kandel S, Heer J, Plaisant C, Kennedy J, van Ham F, Riche NH, Weaver C, Lee B, Brodbeck D, Buono P. Research directions in data wrangling: visualizations and transformations for usable and credible data. Inf Vis. 2011;10(4):271–88. https://doi.org/10.1177/1473871611415994.
14. Dasu T, Johnson T. Exploratory data mining and data cleaning. 1st ed. New York: Wiley; 2003.
15. Abedjan Z, Chu X, Deng D, Fernandez RC, Ilyas IF, Ouzzani M, Papotti P, Stonebraker M, Tang N. Detecting data errors: where are we and what needs to be done? Proc VLDB Endow. 2016;9(12):993–1004. https://doi.org/10.14778/2994509.2994518.
16. Hellerstein JM. Quantitative data cleaning for large databases; 2008.
17. Krishnan S, Haas D, Franklin MJ, Wu E. Towards reliable interactive data cleaning: a user survey and recommendations. In: Proceedings of the workshop on human-in-the-loop data analytics. HILDA '16. ACM, New York, NY, USA; 2016, p. 9–195. https://doi.org/10.1145/2939502.2939511.
18. Polyzotis N, Roy S, Whang SE, Zinkevich M. Data lifecycle challenges in production machine learning: a survey. SIGMOD Rec. 2018;47(2):17–28. https://doi.org/10.1145/3299887.3299891.
19. Ilyas IF, Chu X. Trends in cleaning relational data: consistency and deduplication. Found Trends Databases. 2015;5(4):281–393. https://doi.org/10.1561/1900000045.

20. Park Y, Cafarella MJ, Mozafari B. Visualization-aware sampling for very large databases; 2015. CoRR arXiv :abs/1510.03921.
21. Fisher D. Big data exploration requires collaboration between visualization and data infrastructures. In: HILDA '16 proceedings of the workshop on human-in-the-loop data analytics, San Francisco, California, 26 June–1 July 2016. New York: ACM; 2016. https://dl.acm.org/citation.cfm?id=2939518.
22. Wang Y, Zhong Y, Ma Q, Yang G. Distributed and parallel construction method for equi-width histogram in cloud database. Multiagent Grid Syst. 2017;13(3):311–29.
23. Yang P. Ray dataframes: a library for parallel data analysis. Master's thesis, EECS Department, University of California, Berkeley; May 2018. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-84.html. Accessed 15 Jan 2019.
24. Wickham H, Grolemund G. R for data science: import, tidy, transform, visualize, and model data. 1st ed. Sebastopol: O'Reilly Media Inc; 2017.
25. Wes M. Python for data analysis. 1st ed. Sebastopol: O'Reilly Media Inc; 2012.
26. Chong FT, Heck MJR, Ranganathan P, Saleh AAM, Wassel HMG. Data center energy efficiency: improving energy efficiency in data centers beyond technology scaling. IEEE Design Test. 2014;31(1):93–104. https://doi.org/10.1109/MDAT.2013.2294466.
27. Shvachko K, Kuang H, Radia S, Chansler R. The hadoop distributed file system. In: 2010 IEEE 26th symposium on mass storage systems and technologies (MSST); 2010, p. 1–10. https://doi.org/10.1109/MSST.2010.5496972.
28. Dean J, Ghemawat S. Mapreduce: simplified data processing on large clusters. Commun ACM. 2008;51(1):107–13. https://doi.org/10.1145/1327452.132749.
29. Zaharia M, Chowdhury M, Das T, Dave A. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: NSDI'12 Proceedings of the 9th USENIX conference on networked systems design and implementation; 2012, p. 2. https://doi.org/10.1111/j.1095-8649.2005.00662.x.
30. Zaharia M, Xin RS, Wendell P, Das T, Armbrust M, Dave A, Meng X, Rosen J, Venkataraman S, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I. Apache spark: a unified engine for big data processing. Commun ACM. 2016;59(11):56–65. https://doi.org/10.1145/2934664.
31. Salloum S, Dautov R, Chen X, Peng PX, Huang JZ. Big data analytics on apache spark. Int J Data Sci Anal. 2016;1(3):145–64. https://doi.org/10.1007/s41060-016-0027-9.
32. Bengfort B, Kim J. Data analytics with Hadoop: an introduction for data scientists. Sebastopol: O'Reilly Media; 2016. https://books.google.com.hk/books?id=ou9FDAAAQBAJ.
33. Siuly S, Zhang Y. Medical big data: neurological diseases diagnosis through medical data analysis. Data Sci Eng. 2016;1(2):54–64. https://doi.org/10.1007/s41019-016-0011-3.
34. Jacobs B. White paper: accelerating R analytics with Spark and Microsoft R Server for Hadoop. White Paper; 2016. https://info.microsoft.com/rs/157-GQE-382/images/EN-CNTNT-Whitepaper-Spark-Microsoft-R-Server-Hadoop.pdf. Accessed 17 May 2019.
35. Vargas-Solar G, Zechinelli-Martini JL, Espinosa-Oviedo JA. Big data management: what to keep from the past to face future challenges? Data Sci Eng. 2017;2(4):328–45. https://doi.org/10.1007/s41019-017-0043-3.
36. Dolev S, Florissi P, Gudes E, Sharma S, Singer I. A survey on geographically distributed big-data processing using mapReduce. IEEE Trans Big Data. 2017;99:1. https://doi.org/10.1109/TBDATA.2017.2723473.
37. Salloum S, Huang JZ, He Y. Random sample partition: a distributed data model for big data analysis. IEEE Trans Ind Inf. 2019. https://doi.org/10.1109/TII.2019.2912723.
38. Ci X, Meng X. In: Dong XL, Yu X, Li J, Sun Y (eds) An efficient block sampling strategy for online aggregation in the cloud. Cham: Springer; 2015. p. 362–73.
39. Chaudhuri S, Das G, Srivastava U. Effective use of block-level sampling in statistics estimation. In: Proceedings of the 2004 ACM SIGMOD international conference on management of data. SIGMOD '04. ACM, New York, NY, USA; 2004, p. 287–98. https://doi.org/10.1145/1007568.1007602.
40. Kalavri V, Brundza V, Vlassov V. Block sampling: efficient accurate online aggregation in mapreduce. In: 2013 IEEE 5th international conference on cloud computing technology and science, vol. 1; 2013, p. 250–57. https://doi.org/10.1109/CloudCom.2013.40.
41. Wang Y, Zhong Y, Ma Q, Yang G. Distributed and parallel construction method for equi-width histogram in cloud database. Multiagent Grid Syst. 2017;13(3):311–29. https://doi.org/10.3233/MGS-170273.
42. Wei C, Salloum S, Emara TZ, Zhang X, Huang JZ, He Y. A two-stage data processing algorithm to generate random sample partitions for big data analysis. In: Luo M, Zhang L-J, editors. Cloud Computing—CLOUD 2018. Cham: Springer; 2018. p. 347–64.
43. Emara TZ, Huang JZ. A distributed data management system to support large-scale data analysis. J Syst Softw. 2019;148:105–15. https://doi.org/10.1016/j.jss.2018.11.007.
44. Salloum S, Huang JZ, He Y, Chen X. An asymptotic ensemble learning framework for big data analysis. IEEE Access. 2018;. https://doi.org/10.1109/ACCESS.2018.2889355.
45. Agarwal S, Mozafari B, Panda A, Milner H, Madden S, Stoica, I. Blinkdb: queries with bounded errors and bounded response times on very large data. In: Proceedings of the 8th ACM European Conference on Computer Systems. ACM; 2013, p. 29–42.
46. Krishnan DR, Quoc DL, Bhatotia P, Fetzer C, Rodrigues R. Incapprox: A data analytics system for incremental approximate computing. In: Proceedings of the 25th International Conference on World Wide Web. WWW '16. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland; 2016, p. 1133–44. https://doi.org/10.1145/2872427.2883026.
47. Huang B, Babu S, Yang J. Cumulon: optimizing statistical data analysis in the cloud. In: Proceedings of the 2013 ACM SIGMOD international conference on management of data. SIGMOD '13. ACM, New York, NY, USA; 2013, p. 1–12. https://doi.org/10.1145/2463676.2465273.
48. Budiu M, Isaacs R, Murray D, Plotkin G, Barham P, Al-Kiswany S, Boshmaf Y, Luo Q, Andoni A. Interacting with large distributed datasets using sketch. In: Proceedings of the 16th eurographics symposium on parallel graphics

and visualization. EGPGV '16. Eurographics Association, Goslar Germany, Germany; 2016, p. 31–43. https://doi.org/10.2312/pgv.20161180.
49. Wasay A, Wei X, Dayan N, Idreos S. Data canopy: accelerating exploratory statistical analysis. In: Proceedings of the 2017 ACM international conference on management of data. SIGMOD '17. ACM, New York, NY, USA; 2017, p. 557–72. https://doi.org/10.1145/3035918.3064051.
50. Landset S, Khoshgoftaar TM, Richter AN, Hasanin T. A survey of open source tools for machine learning with big data in the hadoop ecosystem. J Big Data. 2015;2(1):1–36. https://doi.org/10.1186/s40537-015-0032-1.
51. Guha S, Hafen R, Rounds J, Xia J, Li J, Xi B, Cleveland WS. Large complex data: divide and recombine (d&r) with rhipe. Stat. 2012;1(1):53–67. https://doi.org/10.1002/sta4.7.
52. Kleiner A, Talwalkar A, Sarkar P, Jordan MI. A scalable bootstrap for massive data. J R Stat Soc Series B Stat Methodol. 2014;76(4):795–816. https://doi.org/10.1111/rssb.12050.
53. Leys C, Ley C, Klein O, Bernard P, Licata L. Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. J Exp Soc Psychol. 2013;49(4):764–6. https://doi.org/10.1016/j.jesp.2013.03.013.
54. Prokoshyna N, Szlichta J, Chiang F, Miller RJ, Srivastava D. Combining quantitative and logical data cleaning. Proc VLDB Endow. 2015;9(4):300–11. https://doi.org/10.14778/2856318.2856325.
55. Rezig EK, Ouzzani M, Elmagarmid AK, Aref WG. Human-centric data cleaning [vision]. 2017. CoRR arXiv:abs/1712.08971.
56. Doan, A. Human-in-the-loop data analysis: A personal perspective. In: Proceedings of the workshop on human-in-the-loop data analytics. HILDA'18. ACM, New York, NY, USA; 2018, p. 1–116. https://doi.org/10.1145/3209900.3209913.
57. Liu J, Wilson A, Gunning D. Workflow-based human-in-the-loop data analytics. In: Proceedings of the 2014 workshop on human centered big data research. HCBDR '14. ACM, New York, NY, USA; 2014, p. 49–494952. https://doi.org/10.1145/2609876.2609888.
58. Anderson MR, Antenucci D, Cafarella MJ. Runtime support for human-in-the-loop feature engineering system. IEEE Data Eng Bull. 2016;39(4):62–84.
59. Cormode G. Data sketching. Commun ACM. 2017;60(9):48–55. https://doi.org/10.1145/3080008.

Publisher's Note