University of Economics, Prague
Faculty of Informatics and Statistics
Department of Econometrics

Technical Report No. 01/2018

Narrow Big Data in a stream: Computational limitations and regression

Michal Černý

Narrow Big Data in a stream: Computational limitations and regression

Michal Černý

Department of Econometrics
University of Economics, Prague,
Winston Churchill Square 4, 13067 Prague 3
Czech Republic
E-mail: `cernym@vse.cz`

Abstract

We consider the on-line model for a data stream: data points are waiting in a queue and are accessible one-by-one by a special instruction. When a data point is processed, it is dropped forever. The data stream is assumed to be so long that it cannot be stored in memory in full: the size of memory is assumed to be polynomial in the dimension of data, but not the number of observations. This is a natural model for Narrow Big Data. First we prove a negative theorem illustrating that this model leads to serious limitations: we show that some elementary statistics, such as sample quantiles, *cannot* be computed in this model (the proof is based on a Kolmogorov complexity argument). This raises a crucial question which data-analytic procedures can be implemented in the stream data model and which cannot be performed at all, or only approximately (with some loss of information). After the negative results, we turn our attention to several positive results from multivariate linear regression with Narrow Big Data. We prove that least-squares based estimators and regression diagnostic statistics, such as statistics based on the residual sum of squares, can be computed in this model efficiently. The class of statistics efficiently computable in the stream data model also includes two-step procedures involving auxiliary regressions, such as White's heteroscedasticity test of Breusch-Godfrey autocorrelation test (which may be surprising because the procedures, as defined, seem to require a data point to be processed several times). The computation is done exactly: we do not use preprocessing steps involving data compression techniques with information loss (such as sampling or grouping) for a reduction of the size of the dataset.

1 Introduction

Big Data is a hot topic in the current data research and is studied in various contexts in computer science and statistics. The topics covered range from data-driven methods involving clustering [4, 5, 14], detection of changes or failures [3, 8] or detection of frequent events [16] to methods based on advanced statistical theory such as [6, 10]. (This is just a small illustration.)

It is usual to distinguish between Narrow and Wide Big Data. A narrow dataset consists of a huge number of multivariate data points in a moderate dimension, while a wide dataset consists of a moderate number of data points in a huge dimension.

This paper is motivated by the general question *which data-analytic procedures can be performed with Big Data* **efficiently** *and* **exactly** *under strong computational limitations, such as the inability to store the whole dataset in memory at once.* To get a basic insight what kinds of problems are faced, consider the problem of computing the median of a “long” univariate dataset. When the whole dataset cannot be stored in memory, even elementary operations such as sorting become nontrivial. In the stream data model, described in Sections 2.1–2.4, exact computation of median is a good example highlighting the barriers imposed by the computational model. In Section 3 we prove *formally* that the sample median, or any other quantile, cannot be computed exactly. (This does not rule out that an approximate computation is possible; but it will always be with loss of some information.)

This paper is a contribution to the theory of data-analytic computing with narrow datasets. (Wide datasets are not addressed here because they require different methodology with a different computational model than discussed in this paper.) First we introduce a natural computational model for narrow data: it is assumed the data points are supplied in a stream (a queue). We will need a formal definition in terms of Turing machines, but generally it is a kind of model informally referred to as *fully online row-by-row model for data streams* [10].

One of interesting examples, discussed at the 61st World Statistics Congress of the International Statistical Institute[1], is that one flight of a four-engine jet aircraft over the Atlantic produces a data stream of size 600T (and most of the data are dropped). This is a narrow dataset: the data consists of long (high-frequency) series from a moderate number of devices and sensors. Another nice example is the huge data flow from the Large Hadron Collider in CERN — methods processing the data can access the stream by reading data points one by one, but when a data point is processed, it is dropped forever. There is no possibility to store the entire dataset, even with the best contemporary hardware. (Thanks to Karel Ha for this example.)

The stream data model leads to serious computational restrictions. On the one hand, we have complex statistical procedures intrinsically requiring the entire dataset in hand and these procedures cannot be performed in the stream model (unless we reduce the dataset, which leads to a loss of information).

[1]See `isi2017.org` and `www.isi-web.org`.

For example, the maximization of likelihood often requires advanced numerical optimization, which can be hardly thought of without the entire dataset in memory; even linear programming, used e.g. in L_1-norm estimation, is known to be hard-to-decompose into smaller subproblems [7]. On the other hand — and *this is one of the messages of this paper* — it is surprising that there are important procedures, which seem to need the entire dataset in hand and additional memory as huge as the dataset itself, but they can be transformed into a special form suitable for the stream data model.

An example is White's homoscedasticity test (discussed in Section 6.1), or two-step regressions more generally. First, White's test requires to estimate a linear regression model. Then, its residuals are computed. Then, in the second regression, squared residuals are regressed against transformed regressors from the first regression. Finally, the residuals from the auxiliary regression are computed and are used in the F-test or LM-test for the significance of the auxiliary regression. This procedure, as described, requires a data point to be accessed several times. Moreover, two vectors of residuals need to be computed and stored in memory — but this is *impossible* because there are as many residuals as the number of data points. We show how to overcome such limitations and express White's test in a form suitable for stream data. We also study other statistics, such as Breusch-Godfrey test or Jarque-Bera test.

Recall that the usual approach to large datasets is the usage of a preprocessing step which consists in compression of the dataset into a smaller one either by sampling [18] or grouping and replacement of the group by a small vector of suitable characteristics of the group [12, 20]. In this paper we use *no size-reduction procedures*: we deal with exact computations where no information from the dataset is lost.

Finally, it is worth emphasizing that data streaming is a general concept studied across various areas, such as computer science [9], computational statistics [10], data mining [21] and classification [4, 20], as well as in other areas of applied mathematics, such as on-line graphs [17] or on-line scheduling [19].

2 Computational model for Narrow Big Data in a stream

2.1 Formal definition: outline

Let the dataset be organized as a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times p}$ with rows $\boldsymbol{a}_1^{\mathrm{T}}, \dots, \boldsymbol{a}_n^{\mathrm{T}}$. A row is also called *data point*.

Let $\varphi : \mathbb{N} \to \mathbb{N}$ be a nondecreasing function. We say that a function $f(\boldsymbol{A})$ is *$O(\varphi)$-computable in stream*, if the value $f(\boldsymbol{A})$ can be computed under the following restrictions:

(i) The data is available via an instruction GetDataPoint. When it is called for the first time, it returns $\boldsymbol{a}_1$, when it is called for the second time, it returns $\boldsymbol{a}_2$ and so on. When it is called for the $(n+1)$-th time, it returns a special symbol "EndOfData".

(ii) The memory consists of $O(\varphi(p))$ cells. Each cell can store a real number and perform usual arithmetical operations.

If there exists $k \in \mathbb{N}$ (independent of p) such that a function f is $O(p^k)$-computable in stream, we say that f is *polynomially computable in stream.*

Remark 1. The expression "polynomially computable" refers to the fact that the size of memory is restricted by a polynomial in p, the dimension of data. Of course, this is interesting only if $n \gg O(p^k)$; a reader can think of $n = \Omega(2^p)$ for example. This is a formalization of the phenomenon that the memory allows us to store a data point $\boldsymbol{a} \in \mathbb{R}^p$ and perform simple operations, such as inverting $(p \times p)$-matrices (which requires space $O(p^2)$). Thus, the memory is reasonably larger than a single data point, but too small for the storage of the entire dataset or a great portion thereof.

In the context of Narrow Big Data, this model seems to be a reasonable compromise between the restriction on the size of available memory and the ability to compute at least 'something nontrivial'. But even more restrictive stream data models have been studied in literature, see [13].

Remark 2. A stream data model is also known e.g. from on-line quality control [1, 3]. But the context is different: the main restriction is not in the size of memory, but in the costs resulting from a late recognition of a structural break occurring in the on-line observed series.

2.2 A general algorithmic scheme

All algorithms for polynomial stream computing have a similar structure as shown in Algorithm 1. To specify an algorithm, one has to specify three procedures — Initialization, Process($\boldsymbol{a}$) and ComputeOutput — and prove that the overall memory used can be bounded by a polynomial in p.

Algorithm 1 A general algorithmic scheme for in-stream computations

```
{1}  Initialization
{2}  while $\boldsymbol{a}$ := GetDataPoint $\neq$ EndOfData do
{3}      Process($\boldsymbol{a}$)
{4}  end while
{5}  ComputeOutput
{6}  end
```

2.3 A trivial example: one-dimensional data

If $p = 1$, then $\boldsymbol{A} = (a_1, \dots, a_n)^{\mathrm{T}}$ and we have only $O(1)$ memory. The sample mean $\mu := \frac{1}{n}\sum_{i=1} a_i$ can be computed easily:

- Initialization: $n := 0$; $u := 0$;

- Process(a): $n := n + 1$; $u := u + a$;
- ComputeOutput: $\mathit{Output} := u/n$.

The sample variance can be computed in stream by rewriting

$$\frac{1}{n-1}\sum_{i=1}^{n}(a_i - \mu)^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} a_i^2 - \frac{1}{n}\left(\sum_{j=1}^{n} a_j\right)^2\right) \tag{1}$$
$$= \frac{1}{n-1}\left(s_2 - \frac{s_1^2}{n}\right),$$

where $s_\ell = \sum_{i=1}^{n} a^\ell$, $\ell = 1, 2$. The resulting algorithm has the following form:

- Initialization: $n := 0$; $s_1 := 0$; $s_2 := 0$;
- Process(a): $n := n + 1$; $s_1 := s_1 + a$; $s_2 := s_2 + a^2$;
- ComputeOutput: $\mathit{Output} := (s_2 - s_1^2/n)/(n-1)$.

2.4 Refining the computational model: Bounding the size of a memory cell

The definition of the stream computational model from Section 2.1 admits the following kind of cheating. For simplicity assume that $p = 1$ and that all numbers $a_1, \dots, a_n$ are natural. It is possible to encode the entire dataset into a single (huge) natural number. If $(\alpha_{i0}, \alpha_{i1}, \dots, \alpha_{iL_i})$ is the binary encoding of a_i (i.e., $\alpha_{ij} \in \{0, 1\}$ and $a_i = \sum_{j=0}^{L_i} \alpha_{ij} 2^{L_i - j}$), then the natural number with binary encoding

$$1\,\alpha_{10}\,1\,\alpha_{11}\,\cdots\,1\,\alpha_{1L_1}\,0\,\alpha_{20}\,1\,\alpha_{21}\,\cdots\,1\,\alpha_{2L_2}\,0\,\cdots\,1\,\alpha_{nL_n} \tag{2}$$

contains the full dataset (the inserted zero bits serve as delimiters). The current definition of the computational model does not rule out that (2) could be stored in a single memory cell. Of course, this would be an unfair trick. Thus it is necessary to restrict the size of a memory cell.

In addition to (i) and (ii) from Section 2.1, we also add the following requirement:

(iii) The input matrix $\boldsymbol{A} = (a_{ij})_{i=1,\dots,n}^{j=1,\dots,p}$ is rational and the size of each memory cell is bounded by
$$O(Lp \cdot \log^k n) \text{ bits,} \tag{3}$$
where $L = \max_{i,j} \operatorname{bitsize}(a_{ij})$ and $k \geq 1$ is a fixed constant.

Here, bitsize$(\pm\frac{q}{r})$ or a rational number $\pm\frac{q}{r}$ with q, r coprime natural numbers, is the number of bits necessary to write down the sign and the numbers q, r in binary. Clearly, for an integer z, we have bitsize$(z) = O(\log z)$.

Observe that the memory cell of size (3) is sufficient, for example, for the storage of $\sum_{i,j} a_{ij}^\ell$ for $\ell = O(1)$. This property illustrates that the choice (3) is natural: the size of a memory cell is large enough to store some natural quantities and is not unrealistically restrictive, while it rules out the trick (2), which would require $\approx Lpn$ bits.

The stream model as a Turing machine. In theoretical computer science, all algorithms are Turing machines (details can be found e.g. in [2]). The stream data model (i) – (iii) can be formalized as a Turing machine with a read-only input tape equipped with a one-way head, where the data $\boldsymbol{A}$ are stored, and a work tape restricted to

$$O(Lp^k \log^k n) \text{ bits,} \tag{4}$$

where $k \geq 1$ is a fixed constant. This formalization will be useful in Section 3.2. Observe that one call of the GetDataPoint instruction in Algorithm 1 corresponds to reading one row of $\boldsymbol{A}$ from the one-way input tape.

2.5 Remark: Memory-bounded and time-bounded computing

We have not restricted the computation time between two consecutive calls of the GetDataPoint routine. It means that we are admitting also computations with time $\Omega(2^p)$. In a particular situation, it is a matter of context whether the computation time 2^p is considered as manageable or prohibitive. All of the algorithms in this paper use the computation time polynomial in p, so taking a more restrictive model — time-bounded computations instead of memory-bounded computations — does not jeopardize our results from Sections 4–6.

3 Negative results based on Kolmogorov complexity arguments

The main result of this section is that the sample median (or: quantiles in general) cannot be computed in the stream model. The proof is based on Kolmogorov complexity.

First, we shortly summarize the standard definition along the lines of [15]. Recall, informally, that Kolmogorov complexity of a finite 0-1 sequence is the length of the shortest program being able to print the sequence. In a sense, it is the 'best theoretically possible' way of compression of the sequence with no loss of information. For example, a sequence $010101\cdots01$ is very regular and can be printed by a 'short' program, much shorter than the sequence itself. Thus, such a string has low Kolmogorov complexity. But, as it is easy to prove, there are sequences which are 'incompressible', or 'algorithmically random', in the sense that their Kolmogorov complexity is high.

3.1 Notation

Here, a *string* refers to a finite 0-1 sequence. For a string $x \in \{0,1\}^n$, $|x| := n$ denotes its length. A special symbol Λ stands for the empty string (with length 0).

Let a universal Turing machine $U(p, x)$ be fixed. If p, x, y are strings, the relation $U(p, x) = y$ means that the machine U reads the program (formally, another Turing machine) encoded as the string p, simulates its computation with input data x and the computation terminates with output y. When $U(p, \Lambda) = y$, we say that *program p constructs the string y*. The number $|p|$ is referred to as the *length of program p*.

The Kolmogorov complexity $K(y)$ of the string y is defined as

$$K(y) = \min\{|p| : \ U(p, \Lambda) = y\}.$$

A well-known observation shows that there exist string with a high Kolmogorov complexity.

Lemma 3.1 ([15]). *For every n, there is a string $y \in \{0, 1\}^n$ with $K(y) \geq n$.*

Proof. There are 2^n strings of length n, but only $\sum_{k=0}^{n-1} 2^k = 2^n - 1$ programs of length at most $n - 1$, and a program can construct at most one string (a program need not terminate and then it does not construct any string). Thus, at least one string has Kolmogorov complexity at least n. □

3.2 The main negative result: Empirical quantiles are not computable in stream

We restrict ourselves to the computation of the sample median; once it is proven that it is not computable, it is straightforward to see that the argument generalizes to any other quantile as well.

Theorem 3.2. *The sample median is not computable in stream.*

Proof. Consider a one-dimensional ($p = 1$) dataset $a_1, \ldots, a_n$, where all the numbers are natural. Then, $L = O(\max_{i=1,\ldots,n} \log_2 a_i)$. Since $p = 1$, the memory bound (4) of the stream model reduces to

$$O((NL)^k) \text{ bits,} \tag{5}$$

where $k \geq 1$ is a fixed constant and

$$N := \log_2 n.$$

We prove the theorem by contradiction. Assume that there exists a stream algorithm M computing the sample median of $a_1, \ldots, a_n$. Formally, M is a Turing machine with a one-way input tape containing $a_1, \ldots, a_n$ and a work tape which contains a string not longer than $O((NL)^k)$ throughout the computation.

Fix a large enough odd n and a string $y = (y_1, \ldots, y_n)$ with

$$K(y) \geq n, \tag{6}$$

which exists by Lemma 3.1. Define the dataset $a_1, \dots, a_n$ as

$$a_i = \sum_{j=1}^{i} y_j, \quad i = 1, \dots, n.$$

Since $y_j \in \{0, 1\}$, we have

$$0 \le a_1 \le \dots \le a_n \le n, \tag{7}$$

and thus $L = O(\log n) = O(N)$. The memory bound (5) is

$$O((NL)^k) = O(N^{2k}).$$

Now we run the computation of $M(a_1, \dots, a_n)$ and stop just before it calls the GetDataPoint instruction for the $(n+1)$-th time (formally: just before the input tape head attempts to read the first tape cell after the end of the input dataset). The configuration of the machine M is a string s with $|s| \le O(N^{2k})$. [*Proof.* Formally, s is a string containing (i) the string written on the work tape, which has size $O(N^{2k})$, (ii) the head position, which can be written down with $O(k \log N)$ bits, (iii) the internal state of the machine M, which can be written down with $O(1)$ bits. Clearly, $O(N^{2k}) + O(k \log N) + O(1) = O(N^{2k})$.]

Now consider the following algorithm Q. Its body contains the string s, the number n and the text of program M as constants:

```
Program $Q$:
{1} data $n, s, M$
{2} for $i = 1, \dots, n$ do
{3}     $w := (w_1, \dots, w_{n-1}) := (\underbrace{0, 0, \dots, 0}_{n-i \text{ times}}, \underbrace{n, \dots, n}_{i-1 \text{ times}})$
{4}     write $w$ on the input tape of $M$, just on the position to be read
            on its one-way input tape
{5}     simulate the computation of $M$ from the configuration $s$
{6}     $\alpha_i :=$ output of {5}
{7} end for
{8} output $y_1 := \alpha_1$, $y_2 := \alpha_2 - \alpha_1, \dots,$ $y_n := \alpha_n - \alpha_{n-1}$.
```

Step {5} gives the same result as the computation of

$$M(a_1, \dots, a_n, w_1, \dots, w_{n-1}).$$

This is the median of $a_1, \dots, a_n, w_1, \dots, w_{n-1}$, which equals to the i-th smallest element of $a_1, \dots, a_n$ by {3}. By (7), the i-th smallest element is a_i. Thus $\alpha_i = a_i$. It follows that {8} correctly constructs the bitstring $y = (y_1, \dots, y_n)$.

The size of program Q is $O(N^{2k})$ — it just stores the bitstring s of size $O(N^{2k})$, the number n of size $O(N)$ and the remaining code, including the text of program M, has size $O(1)$. Program Q constructs the bitstring y, and thus the size of Q is an upper bound on $K(y)$:

$$K(y) \le O(N^{2k}) = O(\log^{2k} n). \tag{8}$$

If n is sufficiently large, inequalities (6) and (8) are in a contradiction. □

4 Positive results: Linear regression and least squares

Now we turn to positive results in the area of linear regression and regression diagnostic with Narrow Big Data.

Let us assume that the data matrix $\boldsymbol{A} \in \mathbb{R}^{n\times(p+1)}$ has the form

$$\boldsymbol{A} = (\boldsymbol{X}, \boldsymbol{y}),$$

where $\boldsymbol{X} = \mathbb{R}^{n\times p}$ and $\boldsymbol{y} \in \mathbb{R}^n$. The i-th data point — the i-th row of $\boldsymbol{A}$ — has the form $\boldsymbol{a}_i = (\boldsymbol{x}_i^T, y_i)$. This vector is the output of the instruction GetDataPoint.

We consider the linear regression model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon}, \tag{9}$$

where $\boldsymbol{\theta}$ is the vector of regression parameters and the data $(\boldsymbol{X}, \boldsymbol{y})$ are streamed. At this moment we do not need to impose any special assumptions on the disturbances $\boldsymbol{\varepsilon}$. This is because algorithms do not rely on such assumptions. But, of course, once a statistic is computed, additional assumptions are required for the subsequent analysis based on the statistic.

4.1 Least squares and RSS-based statistics

Proposition 4.1. *The OLS estimator $\widehat{\boldsymbol{\theta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$ and the residual sum of squares $RSS = \|\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\theta}}\|_2^2$ are $O(p^2)$-computable in stream.*

Proof. The number n of observations can be computed easily (as in Section 2.3).

In the Initialization step we put $\boldsymbol{J} := \boldsymbol{0}_{p\times p}$, $\boldsymbol{v} := \boldsymbol{0}_{p\times 1}$ and $Y := 0$.

In the Process$(\boldsymbol{x}, y)$ step we update $\boldsymbol{J} := \boldsymbol{J} + \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$, $\boldsymbol{v} := \boldsymbol{v} + y \cdot \boldsymbol{x}$ and $Y := Y + y^2$. At the end of the computation we have the information matrix

$$\boldsymbol{J} = \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$$

and

$$\boldsymbol{v} = \sum_{i=1}^{n} y_i \cdot \boldsymbol{x}_i = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}, \quad Y = \sum_{i=1}^{n} y_i^2.$$

So far we needed $O(p^2)$ memory. The matrix $\boldsymbol{J}^{-1}$ can be computed in $O(p^2)$ memory, too. The ComputeOutput procedure outputs $\widehat{\boldsymbol{\theta}} = \boldsymbol{J}^{-1}\boldsymbol{v}$. For the computation of RSS we can use the expression

$$RSS = (\boldsymbol{y}-\boldsymbol{X}\widehat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{y}-\boldsymbol{X}\widehat{\boldsymbol{\theta}}) = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}-\boldsymbol{y}^{\mathrm{T}}\boldsymbol{X}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = Y-\boldsymbol{v}^{\mathrm{T}}\boldsymbol{J}^{-1}\boldsymbol{v}. \tag{10}$$

□

With $\widehat{\boldsymbol{\theta}}$ and RSS we can compute various derived statistics:

Corollary 4.2. *The following statistics are $O(p^2)$-computable in stream:*

- $\widehat{\sigma}^2 = RSS/(n-p)$ *(the standard estimate of the variance of disturbances),*
- $\widehat{\boldsymbol{\Omega}} := \widehat{\sigma}^2(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1} = \widehat{\sigma}^2\boldsymbol{J}^{-1}$ *(the standard estimate of the covariance matrix of $\widehat{\boldsymbol{\theta}}$),*
- $\widehat{\theta}_i \cdot (\widehat{\boldsymbol{\Omega}}_{ii})^{-1/2}$, $i = 1, \ldots, p$ *(the t-ratios),*
- *the coefficient of determination,*
- *F-tests (e.g. for the hypothesis that all regression coefficients are zero or that all coefficients, except for the intercept, are zero).*

We can summarize: many elementary statistics, which are routinely computed by standard statistical packages, are $O(p^2)$-computable in stream. The conclusion is that *streaming of data makes no serious restriction* (at least in our computational model).

5 Residuals

Analysis of the vector of residuals

$$\boldsymbol{r} := \boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\theta}}$$

is a basic tool for regression diagnostics. In the stream data model, we face the obstacle that $\boldsymbol{r}$ *cannot be stored in memory* — indeed, it has size n, while the memory is assumed to be polynomial in p only. But we have seen that some functions of $\boldsymbol{r}$, such as RSS, can be $O(p^2)$-computed. Here we study some further statistics based on residuals.

5.1 Quadratic forms in $\boldsymbol{r}$ and the DW statistic

Some statistics involve quadratic forms $\boldsymbol{r}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{r}$. (We have already dealt with RSS as the special case with $\boldsymbol{Q} = \boldsymbol{I}$). In general, the matrix $\boldsymbol{Q} = (Q_{ij})$ can be assumed to be lower-triangular. Then we can write

$$\begin{aligned}\boldsymbol{r}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{r} &= (\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\theta}})^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\theta}}) = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{y} - 2\boldsymbol{y}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{X}\widehat{\boldsymbol{\theta}} + \widehat{\boldsymbol{\theta}}^{\mathrm{T}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{X}\widehat{\boldsymbol{\theta}} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{i} Q_{ij}y_iy_j - 2\left(\sum_{i=1}^{n}\sum_{j=1}^{i} Q_{ij}y_i\boldsymbol{x}_j^{\mathrm{T}}\right)\widehat{\boldsymbol{\theta}} + \widehat{\boldsymbol{\theta}}^{\mathrm{T}}\underbrace{\left(\sum_{i=1}^{n}\sum_{j=1}^{i} Q_{ij}\boldsymbol{x}_i\boldsymbol{x}_j^{\mathrm{T}}\right)}_{(\star)}\widehat{\boldsymbol{\theta}}.\end{aligned}$$

As long as the matrix $\boldsymbol{Q}$ has nonzero entries only on the main diagonal and in $O(p)$ diagonals below the main diagonal, then the expression $(\star)$ can be computed in stream in memory $O(p^2)$: it suffices to remember last $O(p)$ data points from the stream and not more (i.e., it suffices to use a sliding window with length $O(p)$). A similar argument holds true for the computation of $\sum_{i\geq j} Q_{ij}y_iy_j$ and $\sum_{i\geq j} Q_{ij}y_i\boldsymbol{x}_j^{\mathrm{T}}$. Thus we have proved:

Lemma 5.1. *Let $\boldsymbol{Q}$ be a lower diagonal matrix with nonzero entries only on the main diagonal and $O(p)$ diagonals below the main diagonal. Then the quadratic form $\boldsymbol{r}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{r}$ is $O(p^2)$-computable in stream.*

The Durbin-Watson (DW) statistic for testing the presence of $AR(1)$ process in disturbances is a nice example: denoting $\boldsymbol{r} = (r_1, \ldots, r_n)^{\mathrm{T}}$, the DW statistic can be written as

$$DW = \frac{1}{RSS}\sum_{i=2}^{n}(r_i - r_{i-1})^2 = \frac{1}{RSS}\left(\sum_{i=1}^{n-1} r_i^2 + \sum_{i=2}^{n} r_i^2 - 2\sum_{i=2}^{n} r_i r_{i-1}\right) = \frac{\boldsymbol{r}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{r}}{RSS}$$

with

$$\boldsymbol{Q} = \begin{pmatrix} 1 & & & & & \\ -2 & 2 & & & & \\ & -2 & 2 & & & \\ & & \ddots & \ddots & & \\ & & & -2 & 2 & \\ & & & & -2 & 1 \end{pmatrix}.$$

Matrix $\boldsymbol{Q}$ satisfies the assumptions of Lemma 5.1. We have:

Proposition 5.2. *The DW statistic is $O(p^2)$-computable in stream.*

Remark 3. Similar considerations hold true also for other statistics based on quadratic forms of residuals, such as Chow's test for the equality of regression parameters between two groups of data. Here we should assume that a data point has the form $(\boldsymbol{x}_i^{\mathrm{T}}, y_i, B_i)$, where B_i is an indicator whether the i-th point belongs to the first or the second group. Then, Chow's statistic has the form

$$\frac{n-2p}{p} \cdot \frac{RSS_{\text{AllData}} - RSS_{\text{Group1}} - RSS_{\text{Group2}}}{RSS_{\text{Group1}} + RSS_{\text{Group2}}},$$

which can be $O(p^2)$-computed in stream.

A negative example: a changepoint version of Chow's statistic. In some situations the stream computation seems to be hard. Assume that the division of data into groups 1 and 2 is not known. Assume further that there exists an index k, called *changepoint*, such that Group 1 consists of data with indices $1, \ldots, k-1$ and Group 2 consists of data with indices $k, k+1, \ldots, n$. If k is unknown, Chow's statistic is often reformulated [22] to the form

$$\frac{n-2p}{p} \cdot \max_k \frac{RSS_{1:n} - RSS_{1:k} - RSS_{k+1:n}}{RSS_{1:k} + RSS_{k+1:n}}, \tag{11}$$

where $RSS_{j:\ell}$ is the OLS-residual sum of squares from regression $y_i = \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\theta} + \varepsilon_i$, $i = j, j+1, \ldots, \ell$. (And the argmax of (11) estimates k.) This is another example where the stream computational model seems to be significantly restrictive.

Remark 4. Generally, matrix $\boldsymbol{Q}$ has size n^2 and it cannot be stored in memory. It must be represented implicitly. We tacitly assumed that the function $(i, j) \mapsto Q_{ij}$ can be evaluated in the available memory.

5.2 Higher moments and Jarque-Bera statistic

The Jarque-Bera test for normality of residuals is based on the statistic

$$JB = \frac{n}{6}\left[\frac{\mu_3^2}{\mu_2^3} + \frac{1}{4}\left(\frac{\mu_4}{\mu_2^2} - 3\right)^2\right],$$

where

$$\mu_\ell = \frac{1}{n}\sum_{i=1}^{n} r_i^\ell. \tag{12}$$

(We use the original version from [11], although other variants of the test have been described in literature.) Obviously, JB is computable in stream if μ_3 and μ_4 are computable in stream (recall that $n\mu_2 = RSS$ and we have shown how to compute it in memory $O(p^2)$). In general, μ_ℓ is not polynomially computable in stream if ℓ is unbounded (more precisely: we don't have an algorithm for evaluation of the function $(\boldsymbol{X}, \boldsymbol{y}, \ell) \mapsto \mu_\ell$ using $O(p^k)$ memory, where k would be a fixed constant independent of ℓ). But, as long as

$$\ell = O(1), \tag{13}$$

μ_ℓ can be computed polynomially.

Lemma 5.3. *The statistic μ_ℓ is $O(p^\ell)$-computable in stream.*

The main result of this section follows:

Proposition 5.4. *The JB statistic is $O(p^4)$-computable in stream.*

Proof of Lemma 5.3. Let

$$\mathbb{K} = \left\{ (k_0, k_1, \ldots, k_p) \in \{0, 1, \ldots, \ell\}^{p+1} \;\middle|\; \sum_{j=0}^{p} k_j = \ell \right\}.$$

Recall that, for $k \in \mathbb{K}$, the multinomial coefficient is defined as

$$\binom{\ell}{k} = \frac{\ell!}{\prod_{j=0}^{p} k_j!}$$

and that Multinomial Theorem is the equality

$$\left(\sum_{j=0}^{p} \xi_j\right)^\ell = \sum_{k\in\mathbb{K}} \binom{\ell}{k} \prod_{j=0}^{p} \xi_j^{k_j}. \tag{14}$$

We apply (14) to μ_ℓ:

$$\begin{aligned}
\mu_\ell &= \frac{1}{n}\sum_{i=1}^{n} r_i^\ell = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^{\mathrm{T}}\widehat{\boldsymbol{\theta}})^\ell = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\widehat{\theta}_j\right)^\ell \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{k\in\mathbb{K}}\binom{\ell}{k} y_i^{k_0}\cdot(-x_{i1}\widehat{\theta}_1)^{k_1}\cdots(-x_{ip}\widehat{\theta}_p)^{k_p} \\
&= \frac{1}{n}\sum_{k\in\mathbb{K}}\binom{\ell}{k}\widehat{\theta}_1^{k_1}\cdots\widehat{\theta}_p^{k_p}\underbrace{\sum_{i=1}^{n} y_i^{k_0}(-x_{i1})^{k_1}\cdots(-x_{ip})^{k_p}}_{=:P_k}. \qquad (15)
\end{aligned}$$

For each $k \in \mathbb{K}$, the value P_k can be computed in stream and stored in a single memory cell. Thus the memory required is $O(|\mathbb{K}|)$.

A counting argument shows that $|\mathbb{K}| = \binom{p+\ell}{p}$; here, $\binom{\ell+p}{p}$ is the ordinary binomial coefficient. Using (13) we can estimate

$$\begin{aligned}
|\mathbb{K}| = \binom{p+\ell}{p} &= \frac{(p+\ell)(p+\ell-1)\cdots(p+1)p!}{p!\ell!} \\
&\leq (p+\ell)(p+\ell-1)\cdots(p+1) \leq (p+\ell)^\ell = O(p^\ell).
\end{aligned}$$

Thus the memory required for the storage of $(P_k)_{k\in\mathbb{K}}$ is $O(p^\ell)$.

At the end of the stream computation, all numbers P_k with $k \in \mathbb{K}$ are available. We can evaluate μ_ℓ using the expression (15), recalling that $\widehat{\theta}_1, \ldots, \widehat{\theta}_n$ are available by Proposition 4.1 in additional $O(p^2)$ space. $\square$

6 Procedures involving auxiliary regressions

We will inspect two important procedures — White's heteroscedasticity test and Breusch-Godfrey autocorrelation test — as representatives of the class of procedures sharing the following common structure:

- *Step I (basic regression)*: the regression
$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \qquad (16)$$
is estimated using OLS and the vector of residuals $\boldsymbol{r}$ is computed;
- *Step II (auxiliary regression)*: another regression
$$\boldsymbol{v} = \boldsymbol{Z}\boldsymbol{\psi} + \boldsymbol{\nu} \qquad (17)$$
is estimated using OLS, where $\boldsymbol{v}$ and $\boldsymbol{Z}$ depend on $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{r})$. Then, the overall significance of the auxiliary regression is tested by the F-test or LM-test, which — in effect — requires stream computation of the residual sum of squares from the auxiliary regression. This sum is denoted by RSS_{aux}.

Such procedures look "sequentially" — first, one needs to compute the residuals $\boldsymbol{r}$ from Step I and then perform the auxiliary regression (17), where $\boldsymbol{r}$ or simple functions of $\boldsymbol{r}$ serve as input data. However, in the stream data model, this cannot be done directly because $\boldsymbol{r}$ cannot be stored in memory the size of which is bounded by a polynomial in p.

To avoid confusion, we still assume that the instruction GetDataPoint returns a row of $(\boldsymbol{X}, \boldsymbol{y})$, which is the vector $(\boldsymbol{x}_i^{\mathrm{T}}, y_i)$ of size $p + 1$, and that the memory is bounded by a polynomial in p, the number of regressors in (16). *All computations necessary for the auxiliary regression must be done within these limitations.*

6.1 White test

Recall that White's heteroscedasticity test works as follows: after estimation of the basic regression (16) and computation of its residuals $\boldsymbol{r}$, in the auxiliary regression (17) we set

$$\boldsymbol{v} = (r_1^2, \ldots, r_n^2)^{\mathrm{T}} \tag{18}$$

and construct the design matrix $\boldsymbol{Z}$ involving the intercept, the regressors from $\boldsymbol{X}$, their squares and cross terms. (In fact, White's auxiliary regression seeks for the dependence of the squared residuals from Step I on low-order polynomials of $\boldsymbol{X}$-regressors.) For example, if $\boldsymbol{x}_i^{\mathrm{T}} = (x_{i1}, \ldots, x_{ip})$ and the basic regression (16) does not involve the intercept, then $\boldsymbol{z}_i^{\mathrm{T}}$, the i-th row of $\boldsymbol{Z}$, has the form

$$\boldsymbol{z}_i^{\mathrm{T}} = (1;\ x_{i1} \ldots, x_{ip};\ x_{i1}^2, \ldots, x_{ip}^2;\ x_{i1}x_{i2}, \ldots, x_{i,p-1}x_{ip}).$$

Matrix $\boldsymbol{Z}$ has n rows. If q denotes the number of columns of $\boldsymbol{Z}$, we have

$$q = O(p^2).$$

For the significance test of the auxiliary regression (17) it suffices to compute RSS_{aux}, the OLS-based residual sum of squares from (17). We call this number as *White's statistic.*[2]

Proposition 6.1. *White's statistic is $O(p^4)$-computable in stream.*

Proof. We need to compute

$$RSS_{\mathrm{aux}} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{v}^{\mathrm{T}}\boldsymbol{Z}(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{v}; \tag{19}$$

we used the same form as in (10). By (12) and (18) we have

$$\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v} = \sum_{i=1}^{n} r_i^4 = n\mu_4,$$

[2]Recall that significance tests compare the unrestricted residual sum of squares RSS_{aux} to the restricted residual sum of squares. For simplicity of presentation, we describe only the stream computation of the unrestricted sum of squares. The stream computation of restricted *RSS* is straightforward — it is a less general problem because the restricted model results from the unrestricted model by deleting some regressors.

which is computable in stream in space $O(p^4)$ by Lemma 5.3.

The matrix $\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z} = \sum_{i=1}^{n} \boldsymbol{z}_i \boldsymbol{z}_i^{\mathrm{T}}$ can be computed in stream in space $O(p^4)$, because $\boldsymbol{z}_i$ can be directly computed from $\boldsymbol{x}_i$ in space $O(p^2)$. In space $O(p^4)$ we can also compute $(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}$.

To complete the stream evaluation of (19), recall that the OLS-estimates $\widehat{\boldsymbol{\theta}} = (\widehat{\theta}_1, \ldots, \widehat{\theta}_p)^{\mathrm{T}}$ of (16) can be $O(p^2)$-computed in stream. It remains to evaluate

$$\begin{aligned}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{v} &= \sum_{i=1}^{n} r_i^2 \cdot \boldsymbol{z}_i = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\widehat{\theta}_j \right)^2 \cdot \boldsymbol{z}_i \\ &= \sum_{i=1}^{n} y_i^2 \boldsymbol{z}_i + \sum_{j=1}^{p} \widehat{\theta}_j^2 \sum_{i=1}^{n} x_{ij}^2 \boldsymbol{z}_i - 2\sum_{j=1}^{p} \widehat{\theta}_j \sum_{i=1}^{n} y_i x_{ij} \boldsymbol{z}_i + 2\sum_{j<k} \widehat{\theta}_j \widehat{\theta}_k \sum_{i=1}^{n} x_{ij} x_{ik} \boldsymbol{z}_i.\end{aligned} \tag{20}$$

There are $O(p^2)$ terms $\sum_i y_i^2 \boldsymbol{z}_i$, $\sum_i x_{ij}^2 \boldsymbol{z}_i$, $\sum_i y_i x_{ij} \boldsymbol{z}_i$, $\sum_i x_{ij} x_{ik} \boldsymbol{z}_i$ which are in-stream computable. Each of the terms has size $O(p^2)$ (because the size of $\boldsymbol{z}_i$ is $O(p^2)$) and the overall required memory is $O(p^4)$. □

Remark 5. A similar method works for some related statistics. For example, White's heteroscedasticity-weighted estimator of $\boldsymbol{\theta}$ has the form

$$\widehat{\boldsymbol{\theta}} = \left(\sum_{i=1}^{n} r_i^2 \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \right)^{-1} \left(\sum_{i=1}^{n} r_i^2 y_i \boldsymbol{x}_i \right)$$

and White's heteroscedasticity-consistent estimator of the covariance matrix $\boldsymbol{\Omega}$ of $\widehat{\boldsymbol{\theta}}$ has the form

$$\widehat{\boldsymbol{\Omega}} = \frac{n}{n-p} (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1} \left(\sum_{i=1}^{n} r_i^2 \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} \right) (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}.$$

The crucial term $\sum_{i=1}^{n} r_i^2 \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}$ can be evaluated in stream in a manner similar to the computation (20) of White's statistic. Observe that this is a special case of Weighted Least Squares, where *the weights are computable in stream.*

6.2 Breusch-Godfrey test

Recall that Breusch-Godfrey (BG) test is a test for the presence of $AR(s)$-process in the disturbances of the basic model (16). Here it is assumed that the lag s satisfies

$$s = O(p), \tag{21}$$

or a reader can simply think of $s = O(1)$. Again, let $\boldsymbol{r} = (r_1, \ldots, r_n)^{\mathrm{T}}$ denote the vector of OLS-based residuals from (17). BG's auxiliary regression (17) has the form

$$\boldsymbol{v} = (r_{s+1}, r_{s+2}, \ldots, r_n)^{\mathrm{T}}, \quad \boldsymbol{Z} = (\boldsymbol{X}_0, \boldsymbol{R})$$

with

$$\boldsymbol{X}_0 = \begin{pmatrix} \boldsymbol{x}_{s+1}^{\mathrm{T}} \\ \boldsymbol{x}_{s+2}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{x}_{n}^{\mathrm{T}} \end{pmatrix}, \quad \boldsymbol{R} = \begin{pmatrix} r_1 & r_2 & \dots & r_s \\ r_2 & r_3 & \dots & r_{s+1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n-s} & r_{n-s+1} & \dots & r_{n-1} \end{pmatrix}.$$

The residual sum of squares from (17), denoted by RSS_{aux} again, is called as *BG-statistic* (see footnote 1 from page 15).

Proposition 6.2. *BG statistic is $O(p^4)$-computable in stream.*

Proof. We have

$$RSS_{\mathrm{aux}} = \boldsymbol{v}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{v}^{\mathrm{T}}\boldsymbol{Z}(\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z})^{-1}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{v}.$$

The term $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{v}$ is a quadratic form in $\boldsymbol{r}$, which can be evaluated by Lemma 5.1.

The matrix $\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}$ has size $O(p^2)$ by (21) and its inverse can be computed in memory $O(p^2)$, too. To compute $\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}$ in stream, we write

$$\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{X}_0^{\mathrm{T}}\boldsymbol{X}_0 & \boldsymbol{X}_0^{\mathrm{T}}\boldsymbol{R} \\ \boldsymbol{R}^{\mathrm{T}}\boldsymbol{X}_0 & \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R} \end{pmatrix}.$$

Each entry of $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$ is a quadratic form in $\boldsymbol{r}$, which can be $O(p^2)$-computed by Lemma 5.1. (Observe that (21) implies that the assumption of Lemma 5.1 is satisfied.) We need memory $O(p^2)$ per element of $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}$, and thus the overall memory is $O(p^4)$. Each entry of $\boldsymbol{R}^{\mathrm{T}}\boldsymbol{X}_0$ is a linear function in $\boldsymbol{r}$, which is in-stream computable and the overall memory does not exceed the previously derived bound $O(p^4)$. A similar argument holds true for the computation of $\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{v}$, too. $\square$

7 Conclusions

We have introduced a computational model for multivariate data supplied in a stream, where the memory is restricted to a polynomial in the dimension p of data. So, the memory is sufficient for storage of a data point or a simple function thereof, a polynomially (in p) bounded sliding data window, or a matrix of size polynomially bounded in p. This model seems to be natural for Narrow Big Data: the memory is reasonably larger than a single data point and is sufficient for performing operations with small objects. But the dataset of size n cannot be stored in general. We asked a question how restrictive this model is for computing least-squares based regression and related diagnostic statistics based on regression residuals. (Recall that the vector of residuals has size n and thus it cannot be stored in memory.) We showed that many statistics, routinely computed by statistical packages, are efficiently computable in this model and that streaming of data does not make a serious restriction. We also showed that some two-step procedures, involving an auxiliary regression with residuals from the first-step regression, are computable in this model efficiently, too. Two important representatives are the White test and the Breusch-Godfrey test.

Conversely, we have also discussed some statistics which are hard-to-compute in this model; the examples are quantiles or some max-type statistics of Chow's type used in changepoint analysis. Negative proofs — that a statistic cannot be computed in the stream model — are based on Kolmogorov complexity arguments. This does not preclude that such statistics could not be computed approximately, e.g. with a preprocessing size-reduction step such as sampling. The negative proofs tell us that the computation is *in principle* possible *only* with some loss of information from the original dataset.

Acknowledgment. The work was supported by the Czech Science Foundation under Grant 16-00408S.

References

[1] J. Antoch and D. Jarušková (2002). On-line statistical process control. In: C. Lauro, J. Antoch, V. Vinzi and G. Saporta (Eds.), *Multivariate Total Quality Control: Foundations and Advances.* Book series: Constributions to Statistics, Physica Verlag HD, 87–124.

[2] S. Arora and B. Barak (2009). *Computational Complexity: A Modern Approach.* Cambridge University Press.

[3] D. Bodenham and N. Adams (2017). Continuous monitoring for changepoints in data streams using adaptive estimation. *Statistics & Computing* **27** (5), 1257–1270.

[4] F. Cao, M. Estert, W. Qian and A. Zhou (2006). Density-cased clustering over an evolving data stream with noise. In: J. Ghost, D. Lambert, D. Skillicorn and J. Srivastava (Eds.), *Proceedings of the 2006 SIAM Conference on Data Mining*, SIAM, 328–339.

[5] F. Cao, J. Z. Huang, J. Liang (2014). Trend analysis of categorical data streams with a concept change method. *Information Sciences* **276**, 160–173.

[6] T. Cipra and R. Romera (1991). Robust Kalman filter and its applications in time series analysis. *Kybernetika* **27**, 481–494.

[7] D. Dobkin, R. Lipton and S. Reiss (1979). Linear programming is Log-Space hard for P. *Information Processing Letters* **8** (2), 96–97.

[8] A. Forestiero (2016) Self-organizing anomaly detection in data streams. *Information Sciences* **373**, 321–336.

[9] M. Garofalakis, J. Gehrke and R. Rastogi (2016). *Data Stream Management: Processing High-Speed Data Streams.* Book Series: Data-Centric Systems and Applications, Spinger-Verlag Berlin Heidelberg.

[10] L. Ippel, M. Kaptein and J. Vermunt (2016). Estimating random-intercept models on data streams. *Computational Statistics & Data Analysis* **104**, 169–182.

[11] C. Jarque and A. Bera (1987). A test for normality of observations and regression residuals. *International Statistical Review* **55** (2), 163–172.

[12] N.-A. Khumbah and E. Wegman (2003). Data compression by geometric quantization. In: M. Akritas and D. Politis (Eds.), *Recent Advances and Trends in Nonparametric Statistics*, Elsevier, 35–46.

[13] L. Kontorovich (2012). Statistical estimation with bounded memory. *Statistics & Computing* **22** (5), 1155–1164.

[14] S. Laohakiat, S. Phimoltares amd C. Lursinsap (2017). A clustering algorithm for stream data with LDA-based unsupervised localized dimension reduction. *Information Sciences* **381**, 104–123.

[15] M. Li and P. Vitányi (2008). *An Introduction to Kolmogorov Complexity and Its Applications* (edition three). Texts in Computer Science, Springer.

[16] H. Lin, S. Wu, L Hou, N. M. Kou, Y. Gao and D. Lu (2018). Finding the hottest item in data streams. *Information Sciences* **430–431**, 314-330.

[17] A. McGregor (2014). Graph stream algorithms: A survey. *ACM SIGMOD Record* **43** (1), 9–20.

[18] B.-H. Park, G. Ostrouchov and N. Samatova (2007). Sampling streaming data with replacement. *Computational Statistics & Data Analysis* **52** (2), 750–762.

[19] J. Sgall (1998). On-line scheduling. *Lecture Notes in Computer Science* **1442**, 196–231.

[20] P. Steiner and M. Hudec (2007). Classification of large data sets with mixture models via sufficient EM. *Computational Statistics & Data Analysis* **51** (11), 5416–5428.

[21] C.-W. Tsai, C.-F. Lai, H.-C. Chao and A. Vasilakos (2015). Big data analytics: a survey. *Journal of Big Data* 2:21, 1–32.

[22] K. Worsley (1983). Testing for a two-phase multiple regression. *Technometrics* **25**, 35–42.