Master Thesis - Computer Science
University of Twente
Faculty of EEMCS
Databases Group

On a Design Space for Aggregating over Sliding Windows on a Stream

Michael Meijer
May 30, 2011

Committee:

dr. A. Wombacher	(UT/DB)
dr. M.M. Fokkinga	(UT/DB)
ing. B. Stemerdink	(Info Support B.V.)
drs. E. van Alebeek	(Info Support B.V.)

Dedication and Acknowledgments

I would like to dedicate this thesis to my parents Arie and Wilma. Over the past few years my life changed a lot. Without their support during that period, I would not have come this far.

This thesis is the final part of my Computer Science master at the University of Twente. From a variety of interesting courses, I learned a lot during the master. I had the honor to do my research on a very interesting topic at Info Support B.V., Veenendaal. Its a pleasure to thank all the people that supported me one way or another during the master and in particular the research leading to this thesis.

I am grateful to my supervisors dr. Andreas Wombacher and dr. Maarten Fokkinga from the University of Twente for their great support throughout the research. Their ever critical view on my work and suggestions to improve it were very helpful. My thanks must go to Juan Amiguet, a PhD student at the University of Twente, for some suggestions on an early draft of my work. I would like to thank my supervisors from Info Support: ing. Bas Stemerdink and drs. Erma van Alebeek for reviewing my work and providing me with the means to conduct the research. I thank my colleagues at Info Support, in particular Martijn Adolfsen, for the great fun we had.

At home, my lovely girlfriend Alina and wonderful children Max and Jasmijn helped me to relax and motivate me to keep on going at times I needed it the most.

Michael Meijer
May, 2011

Abstract

On-line, continuous processing over streaming data challenges performance: processing time and memory space usage. This research is about the best performing way to compute an aggregate function over sliding windows containing the most recent data of a stream. The performance depends on the output semantics: input, aggregate function, accuracy requirements, retention as the quantity of elements aggregated and slide as the quantity of elements to move over the stream. Two approaches are identified, that under the same output semantics, differ in the way sliding windows are materialized. The instantaneous approach takes a window covering the retention and computes the aggregate function over the window upon every slide, independent of previous windows. The incremental approach has a synopsis structure covering the retention, updated by consecutive disjoint batches covering a slide.

A proposed theoretical model relates the approaches to processing time and memory space usage. Scalability is assessed by instantiating that model with the computational complexity of the approaches. It is shown that the incremental approach with a trivial synopsis storing the entire retention can have processing time benefits over the instantaneous approach. Moreover it seems approximations of aggregate functions are the only way to reduce memory space usage when sliding one element at a time. Properties of aggregates from literature are related to the approaches and theoretical model. Aggregates with the properties of a priori bounded space and incremental / decremental functions are called bounded differential aggregates. Under the incremental approach they require constant processing time and can significantly reduce memory space usage in practice when the slide is sufficiently large. By instantiating the theoretical model with empirical approximations of performance, approaches can be compared by their performance in practice. Performance in practice shows differences between approaches belonging to the same complexity class, more average-case like performance and (unexpected) performance side-effects.

A taxonomy is proposed categorizing synopsis structures by support for fixed-size (e.g. count-based) or variable-size (e.g. time-based) retention, exact or approximate aggregates produced and the deterministic or randomized / probablistic result of approximations. A few synopsis structures covering the taxonomy are explored in the context of the selected aggregates variance and quantiles of (most) interest to Info Support. Exact computations are explored by the trivial synopsis that stores the entire retention and panes that exploit a priori space bounds of aggregates for sufficiently large batches. Approximations are explored by sampling, histograms for variance only and summaries for quantiles only.

The selected aggregates under both approaches along with synopsis structures are implemented on the streaming data processing system Microsoft StreamInsight 1.1 of interest to Info Support. A stream of uniform random numbers is used. Accuracy of 95%-100% is of interest to Info Support. The effect of retention is observed by increasing it at a fixed slide. The effect of slide (inversely: overlap) is observed by a fixed retention and increasing slide. The empirical results show the instantaneous approach outperforms the incremental approach when the slide (and batch) is sufficiently large; at about half the retention. Sampling achieves high accuracy and the best performance under the incremental approach. Panes have the best performance when the batches are large enough. Quantile summaries reduce memory space usage significantly, have high accuracy and low processing time. Using SPSS 17 the empirical results are generalized on the environment by regression models for processing time and memory space usage per aggregate, per accuracy requirements and per approach (differentiated by synopsis structure). Violation of the assumptions of homoscedasticity, independence or normality of residuals during analysis, suggests the use of more advanced analysis. Validation of the models on an independent subset of the measurements indicates errors similar to the model errors from analysis.

A design space is proposed that uses the theoretical model to compare the approaches in terms of performance, either by scalability or by empirical approximations of performance in practice, and is augmented with guidelines derived during the research. It indicates the best performing approach regarding how to compute an aggregate function over sliding windows, given output semantics. By exploring other aggregates and synopsis structures, the design space can be extended.

Contents

List of Abbreviations

B	Number of elements in a batch / small window, see chapters 2 and 3.
$BLUE$	Best linear unbiased estimator (BLUE) regarding regression coefficients, see chapter 7.
$1-\delta$	Success probability lower bound on achieving the relative error upper bound (ϵ), see chapter 4.
ϵ	Relative error upper bound on approximations, see chapter 4.
MAE_v	Mean absolute error from the validation of a regression model, see 7.
$O(\cdot)$	Upper bound on the growth rate, see chapter 3.
OLS	Ordinary least squares, used for regression analysis in chapter 7.
R	Number of elements in the retention, see chapters 2 and 3.
R^2	Coefficient of determination, the proportion of variance in the dependent variable explained by the independent variables in regression models, see chapter 7.
$RMSE$	Root mean squared error known as the standard error of the estimate of a regression model, see 7.
$RMSE_v$	Root mean squared error from the validation of a regression model, see 7.
S	Number of elements in the slide, see chapters 2 and 3.
$S(\cdot)$	Memory space usage, e.g. computational space complexity or empirical approximation, see chapters 3, 6 and 7.
$S(R)$	Memory space usage with respect to the retention in the instantaneous approach, see chapters 3, 6 and 7.
$S(R, B)$	Memory space usage with respect to the synopsis over the retention and batch in the incremental approach, see chapters 3, 6 and 7.
SSE	Sum of squared residuals, minimized in ordinary least squares (OLS) regression, see chapter 7.
$T(\cdot)$	Processing time, e.g. computational time complexity or empirical approximation, see chapters 3, 6 and 7.
$T(R)$	Processing time with respect to the retention in the instantaneous approach, see chapters 3, 6 and 7.
$T(R, S, B)$	Processing time with respect to the retention, slide and batch in the incremental approach, see chapters 3, 6 and 7.
$T_q(R)$	Processing time to query the synopsis over the retention in the incremental approach, see chapters 3, 6 and 7.
$T_u(R)$	Processing time to update the synopsis over the retention for a new element in the incremental approach, see chapters 3, 6 and 7.
$\Theta(\cdot)$	Upper and lower bound on the growth rate, see chapter 3.
$\Omega(\cdot)$	Lower bound on the growth rate, see chapter 3.
WLS	Weighted least squares, used for regression analysis in chapter 7.

Chapter 1

Introduction

At the software company Info Support, the Business Intelligence unit has thought about streaming data processing, applied next to traditional processing with a data warehouse. An interesting aspect is the processing of (aggregate) functions over some recent yet finite part of the stream, called a window, that slides when new data arrives. This is the main topic of this research.

Surveys, general information on streaming data processing and systems are treated in various research [10, 18, 19, 21, 22, 36, 53, 58]. Some of the recurrent requirements / issues are:

- On-line, continuous and bursty arrival of data as a potentially unbounded stream that cannot be persisted in its entirety.
- Performance in terms of processing time (latency) and memory space usage constraints in on-line (and in-memory) processing of the data.
- Working on finite parts of the stream known as windows (e.g. only data from the hour), sliding when new data becomes available and providing periodic results.
- Resource sharing at machine/system, query or operator level.

This chapter starts with a motivational case from Info Support in section 1.1. Next two conceptually different approaches are outlined in section 1.2 to compute the same (aggregate) function over a sliding window. It is to provide an intuition for the approach, while a formal discussion is in chapter 2. The problem statement, research questions and contributions are described in section 1.3. A discussion of the method employed for the research follows in section 1.4. Finally, the structure of this document is outlined in section 1.5.

1.1 Motivational Case

The interest in streaming data processing by Info Support is motivated by a case. In hospitals MRI scanners among others are being used. Such scanners require helium levels, temperature and other status information to be monitored, e.g. to ensure the correct functioning of the scanner. Periodically measurements of helium and temperature are taken and sent to an operational system along with other status information. This is effectively a stream of measurements. The operational system handles data streams of thousands of scanners across hospitals; a number that can grow in a foreseeable future (potentially up to 50000). Periodically the operational system sends data to a backing data warehouse used for analysis. Clearly if the measurements are sent frequently by all scanners, e.g. every minute, the amount of data to process and store rapidly becomes large.

Aggregating voluminous amounts of data is beneficial for reduction or having data at a desired level of granularity [42]. By aggregating directly on the streaming data, both the operational and backing data warehouse have to store and process a lot less data. Moreover, processing on the stream provides preprocessed data early on. Potentially the time to act on incorrect functioning of the scanners can be reduced significantly. For example, while individual helium level measurements might not cause warning, the average helium level over some recent period of time needs to sustain a certain level with limited variance. Perhaps temperature measurements should have at least 99 percent of the measurements at or above some level (the 0.99-quantile should have some acceptable value). Problems can be detected when they happen and maintenance personnel be involved early on. Cost savings can be realized from reduced down time by the early detection of problems. Waiting for the operational system or backing analysis can cost valuable time.

The notion of the last, most recent period or number of measurements is captured by windows of data over a stream. Such windows slide when new data arrives to limit the data to process to the recent past [10]. Such windows are claimed to be natural, well-defined and easily understood. Dealing with (aggregate) functions computed over such sliding windows with performance in mind, is of interest to Info Support to handle cases like above.

1.2 Instantaneous and Incremental Approaches

The on-line and continuous arrival of data as a stream that needs to be processed on-line (and in-memory) by sliding window, is subject to performance constraints in terms of processing time and memory space usage. Let sliding windows be put conceptually as:

- Retention, as the number (or period) of elements over which the (aggregate) function is computed.
- Slide, as the number (or period) of elements by which the retention moves over the stream; it is the inverse of the overlap between consecutive retentions:
  - The newest elements become part of the retention.
  - The oldest elements are removed from the retention; they are expired.

Two conceptually different approaches in computing the same (aggregate) functions over the retention that slides are discussed informally. Their formal definitions can be found in chapter 2. The difference between the approaches is in they way the retention that slides, behaving as a sliding window, is materialized and the (aggregate) function is computed. They have different performance characteristics, however their output is semantically equivalent.

1.2.1 Outline and Example of the Approaches

The two approaches are: the instantaneous and the incremental approach. The instantaneous approach materializes the retention by a large sliding window of data over which to compute the (aggregate) function over all data at once. After sliding, the next large window of data representing the retention is processed without regard of the previous window. Overlapping data is reprocessed. The incremental approach takes consecutive disjoint batches / small windows of data from the stream and maintains a synopsis of the retention; the retention is materialized by the batches and synopsis. For every new batch of data, old data no longer part of the retention is undone from and new data is applied to the synopsis. By accounting for new and expired data in the synopsis, the overlapping part of the retention upon sliding is reused. The synopsis covers the retention and once every one or more batches cover the slide, a value for the (aggregate) function over the retention is queried from or looked up in the synopsis. A synopsis yields aggregates with a certain level of accuracy depending on how well the synopsis reflects the retention. In figure 1.1 the approaches are depicted. It shows a stream with new elements of data as time progresses. The notions of retention, slide / overlap and batch are depicted. The instantaneous approach is shown with a window i moving to $i + 1$. The incremental approach has batches j, $j + 1$, ..., $j + 5$ and a synopsis after $j + 4$ becoming the synopsis after $j + 5$ upon having processed the corresponding batch. Next an example is given to demonstrate the approaches in computing the average.

Example 1.2.1. Assume there is a stream of temperature measurements with a timestamp. Suppose a retention of 10 seconds and a slide of 2 seconds; a sliding window. A 12 second excerpt of the stream is shown below.

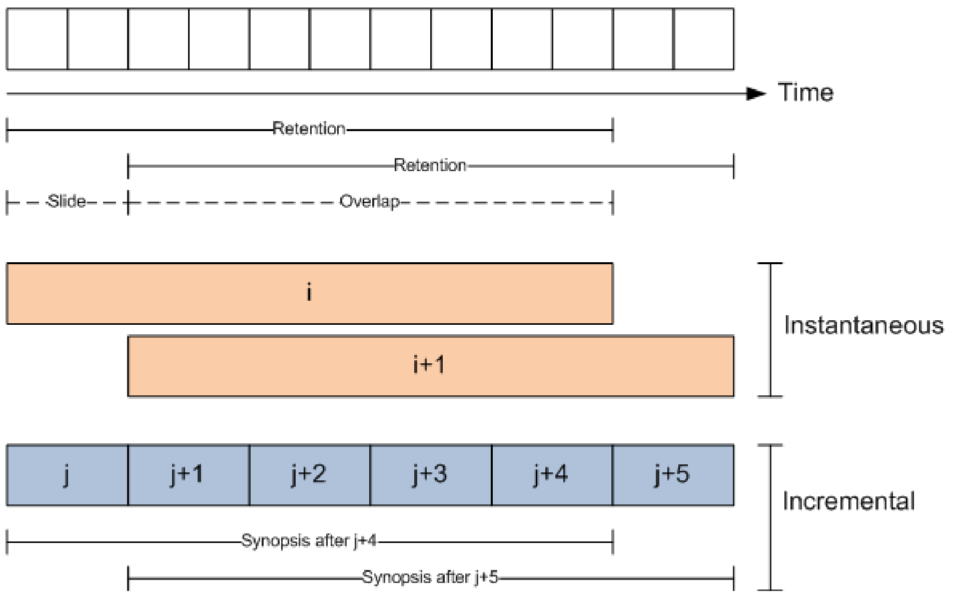


Figure 1.1: Visualization of the instantaneous and incremental approaches. Elements of data from the stream are shown above the time axis (white boxes). The large rectangles represent the sliding windows used in the instantaneous approach to cover the retention. The small rectangles represent the consecutive disjoint batches / small windows of the incremental approach with a synopsis covering the retention. ◦

Timestamp	Temperature
12:00:00	21.0
12:00:01	22.0
12:00:02	23.0
12:00:03	19.0
12:00:04	20.0
12:00:05	24.0
12:00:06	26.0
12:00:07	22.0
12:00:08	22.0
12:00:09	23.0
12:00:10	24.0
12:00:11	20.0

The instantaneous approach computes the average temperature over 10 second retentions by large window, sliding every 2 seconds. The average over the first 10 second retention of the excerpt is: the sum of the temperatures of the interval [12:00:00, 12:00:09] (a large window) divided by the number of measurements in that interval, hence $(21.0 + 22.0 + ... + 22.0 + 23.0)/(1 + 1 + ... + 1 + 1) = 222.0/10 = 22.2$. When the retention slides, it covers the last 10 seconds of the excerpt. Regardless of the previous average computation, the sum and count of temperature measurements is computed over the last 10 seconds, $(23.0 + 19.0 + ... + 24.0 + 20.0)/(1 + 1 + ... + 1 + 1) = 223.0/10 = 22.3$. The instantaneous approach needs to process all measurements in the retention every time it computes the average.

The incremental average temperature requires a synopsis, say a sum and count over the retention. It uses batches of 2 seconds so that a single batch matches precisely the data by which the retention slides. The first batch consists of (12:00:00, 21.0) and (12:00:00, 22.0), the second batch consists of (12:00:02, 23.0) and (12:00:03, 19.0), etc. For simplicity suppose the sum and count initially are 0. After the first batch the sum is 43.0 and count is 2. After the second batch the sum is 85.0 and count is 4. As the fifth batch is processed the sum is 222.0 and count is 10. The synopsis yields an average of $222.0/10 = 22.2$. After the sixth batch is processed, the first batch is undone and the sixth batch applied. Undoing the first batch yields a sum of $222.0 - 43.0 = 179.0$ and a count of $10 - 2 = 8$. Applying the sixth batch yields a sum of $179.0 + 44.0 = 223.0$ and a count of $8 + 2 = 10$. The synopsis yields an average of $223.0/10 = 22.3$. The synopsis reuses the contributions of the second up until the fifth batch. Note that even though the approaches are conceptually different, their output semantics are the same. ◇

1.2.2 Alternative Approaches in Literature

The aforementioned incremental approach is not unique in itself or the only alternative to the instantaneous approach. Some pointers to related literature on incremental processing are provided to the interested reader and put the incremental approach into perspective. A closely related approach is the use of panes [47]. The stream at the pane level is divided in consecutive non-overlapping batches of data, called panes, each of which is aggregated over. The panes making up a window are aggregated to obtain the sliding window aggregate. Properties of aggregates related to this idea are discussed in chapter 3. It also inspires synopsis structures used in this research, as discussed in the corresponding chapter 4.

An alternative way of processing elements of the stream is by assigning a range of identifiers to elements indicating to which windows they belong [48]. An aggregate operator applies an element to the aggregates for the windows whose identifiers match the element's range of window identifiers. The assignment of such identifiers is non trivial. In a sense windows are not materialized, thus preventing a buffer for a window worth of elements and applying a tuple once to all windows it belongs. Rather partial aggregates are buffered for windows and upon the system indicating the end of window extents, the aggregate is output and no longer buffered.

Yet another approach uses a sequence of basic windows, i.e. consecutive disjoint small windows, each maintaining a synopsis structure [66]. Such a sequence covers a sliding window for which a synopsis structure is maintained too. Note the resemblance with the incremental approach where batches / small windows and a synopsis structure are used. The approach is used to optimize the computation of various statistics, in a multi-stream context.

Sharing between operators with the same input stream and aggregate, though different sliding window specifications, is explored in literature [7]. Intervals over a stream are aggregated and used to construct aggregates over larger intervals. The aggregated intervals are shared by windows and a disjoint union of intervals can be used to approximate aggregates for sliding windows. This approach is more flexible than the fixed size batches of the incremental approach. It is used by a synopsis structure in chapter 4.

1.3 Research

This section shortly describes the problem statement and envisaged solution. The main research question and its sub questions follow with pointers to the chapters providing the answers. The section concludes with the main contributions of this research.

1.3.1 Problem Statement

The performance costs, regarding processing time and memory space usage, play a major role in computing (aggregate) functions on sliding windows over data streams. Aggregates are of most interest, as they challenge performance by operating on large amounts of data. For a given stream of data and an aggregate, the choice of how to compute an aggregate function over sliding windows is determined by the performance of the instantaneous and incremental approaches. However performance is affected by the algorithms to compute the aggregate and to maintain synopsis structures with a given accuracy requirement. Moreover the size of the retention and the slide (or inversely: overlap) affect the trade-off regarding reprocessing of large windows by the instantaneous approach versus the overhead of synopsis maintenance per batch / small window in the incremental approach. Furthermore, the performance is affected by characteristics of a streaming data processing system, like its efficiency in handling windows of data or how it optimizes its resource usage. Even the environment, of which the system is a part, can affect performance.

A design space is proposed in chapter 8 that uses the theoretical model of the performance of the approaches in chapter 3 to determine which approach performs best in terms of processing time and memory space usage. It can be used to determine how to compute an aggregate function over sliding windows from the perspective of performance. By the theoretical model approaches can be compared by:

- Scalability in terms of computational complexity of processing time and memory space usage.
- Empirical approximations in terms of processing time and memory space usage approximations based on performance in practice, i.e. in a concrete environment.

The design space is augmented with guidelines derived from (properties of) aggregates, synopsis structures and empirical performance for the approaches. External constraints, e.g. from a business case like prioritizing processing time over memory space usage, can be applied in conjunction with the design space.

1.3.2 Research Questions

The following states the main research question:

How to compute an aggregate function over sliding windows?
Subject to:

- The perspective of performance; a common issue in streaming data processing (as indicated at the beginning of this chapter).
- The instantaneous and incremental approaches of computation (see section 1.2 and chapter 2).
- Bias toward aggregates and synopsis structures of interest to Info Support (see chapters 3 and 4).

The goal is to determine which approach is most performance cost effective, in terms of processing time and memory space usage, in computing an aggregate function over sliding windows given the output semantics: input, aggregate, accuracy requirements, retention and slide. The design space proposed in 8 is the envisaged solution.

To answer the main research question, the following subquestions need to be answered:

- How is the computation of aggregates related to the approaches?

  The goal is to relate the computation of aggregates to the approaches by performance, in terms of processing time and memory space usage, in a theoretical model. Moreover properties of aggregates mentioned in literature are discussed in the context of the approaches. A selection of aggregate functions of (most) interest to Info Support is made. They guide the discussion of synopsis structures and the empirical part of the research. See chapter 3.

- How are synopsis structures related to the incremental approach?

  The goal is to understand what characterizes synopsis structures in the incremental approach and to discuss some synopsis structures related to selected aggregate functions. A taxonomy is proposed and a few synopsis structures for the selected aggregates are examined. They are chosen to cover the taxonomy and provide Info Support with a broad view on the matter to aid in their understanding of streaming data processing. See chapter 4.

- What is the performance of the approaches in practice (empirically)?

  The goal is to get an insight into the performance of the approaches in practice: to observe differences between approaches belonging to the same computational complexity class, observe more average-case like behavior and observe (unexpected) side-effects of performance on a system. The performance in practice (in a concrete environment), is generalized to empirical approximations by regression models. The setup of the empirical measurements is described in chapter 5, the empirical results are presented and discussed in chapter 6 and the regression models in chapter 7.

- What is a design space for computing aggregates under the approaches?

  The goal is to obtain a means to determine which approach has the best performance regarding processing time and memory space usage; referred to as the design space. The theoretical model of chapter 3 is used to compare the performance of the approaches by computational complexity and by empirical approximations. Guidelines derived from aggregates, synopsis structures and empirical performance of the approaches augment the design space. See chapter 8.

1.3.3 Contributions

There are two main stakeholders in this research: the University of Twente/science and the company Info Support. Note that from a business perspective, Info Support is the customer and their interest originates from the business intelligence (BI) unit and application in a BI context. Contribution of the research and its results are not limited to Info Support. From a scientific perspective the following contributions are envisaged:

- The use of the theoretical model of chapter 3 in the design space of chapter 8:
  - To compare the approaches by scalability using computational complexity; a mathematical way independent of the environment (e.g. streaming data processing system).
  - To compare the approaches empirically by regression models of performance in practice.
- Guidelines for computing aggregates regarding the approaches, e.g.:
  - The instantaneous approach has processing time and memory space usage complexity at least linear in the number of elements in the retention (see chapter 3).
  - As the slide (and batch) gets sufficiently large with respect to the retention, the instantaneous approach outperforms the incremental approach regarding processing time and / or memory space usage in practice (see chapter 6 and table 8.1).
  - The trivial synopsis structure (storing each element of the retention) has processing time benefits in practice, if the slide is sufficiently small (see chapter 6).

From a business perspective the contributions are:

- A design space for making decisions in business cases with respect to streaming data aggregates over sliding windows by comparing approaches in terms of performance and using the guidelines. Suppose the motivational case 1.1. The design space can be used to determine which approach to use for the variance of the helium level of MRI scanners over the last 24 hours.
- A means to measure performance in practice and derive empirical approximations by regression models. This can be used to extend the empirical part of the design space to other aggregates and synopsis structures and to tailor it to business cases (e.g. environment configurations).
- Insight into performance in practice of a streaming data processing system; regarding the approaches. The Microsoft StreamInsight v1.1 streaming data processing system was used, which fits into the BI stack of Microsoft products at Info Support.
- A broad coverage of synopsis structures for aggregates of interest to Info Support with literature pointers to extend support to other aggregates of interest.

1.4 Method

This section describes the method used obtain the design space as a means to determine how to compute aggregates (or aggregate functions) over sliding windows over a stream. Chapters providing an in-depth treatment of parts of the method are indicated where appropriate. Whenever a part of the method provides answers to the research question and sub questions, this is indicated as well.

1.4.1 Literature and Theory

Formal semantics of the approaches are described in chapter 2. This is loosely based on literature [6, 14, 46, 48, 52, 56, 58] regarding formal semantics of streaming data processing. The literature is used as a source of inspiration for and extended to the semantics of the approaches. The aim is not to come up with a universal formal model or be complete regarding streaming data processing. The purpose is to provide a common understanding extending upon informal descriptions come across earlier this chapter, in particular regarding the approaches 1.2.

Next a theoretical performance cost model is derived in chapter 3. It relates the computation of aggregates to the approaches in terms of processing time and memory space usage in an abstract way. The theoretical model can be used to reason about the scalability of the approaches by using computational complexity. It can be employed to show if two approaches belong to the same complexity class or whether one bounds the other from above or below. Moreover the effect on scalability by the minimization or maximization of the slide regarding the retention can be explored. Properties of aggregates mentioned in literature [7, 38, 42, 47, 50] are related to the scalability of the approaches, to state propositions on the scalability of aggregates with certain properties. Using the theoretical model with empirical approximations allows the comparison of performance (in practice) of the approaches. Finally some aggregates of (most) interest to Info Support are selected and their instantaneous computation outlined. The selected aggregates guide the discussion on aggregates and the empirical part of the research. The theoretical model answers the first subquestion in section 1.3.2.

The synopsis structures are discussed in chapter 4. Many different kinds of synopsis structures are known from literature [2, 7, 10, 33, 36], however the incremental approach characterizes synopsis structures of interest. A notion of retention must be supported. Moreover expired elements are not explicitly presented as removals from the synopsis structure, hence the synopsis structure must handle them implicitly. Another aspect of synopsis structures is their accuracy. For Info Support synopsis structures with exact aggregates obtained from them or with a bounded error on approximate aggregates obtained from them are of most interest. In literature synopsis structures with a configurable relative error bound (ϵ) can be found. Some are probablistic / randomized by a success probability $(1-\delta)$ of achieving the relative error bound. Some concrete synopsis structures are examined for the selected aggregates in chapter 3 that support retention, (implicit) expiration and accuracy aspects. This provides Info Support with a broad coverage of synopsis structures, rather than an in-depth treatment of a particular structure and its optimizations. The characterization of the synopsis structures by retention, (implicit) expiration and accuracy aspects answers the second subquestion in section 1.3.2.

1.4.2 Empirical Part

Apart from the use of the theoretical model for scalability, it can be used with empirical approximations of processing time and memory space usage to compare the performance of the approaches in practice (i.e. in a concrete environment). Moreover, empirical measurements of performance in practice can provide useful insights because:

- Computational complexity is asymptotic: it has only the characterizing terms without multiplicative constants, is biased toward large inputs and can overstate performance costs by being overly conservative or be observed only in particular circumstances (e.g. specific inputs that occur infrequently in practice).
- Approaches belonging to the same computational complexity are equally scalable, however performance in practice can be clearly different (e.g. different multiplicative constants).
- The performance is affected by interactions with the environment and (unexpected) side-effects, e.g. as a consequence of the design of the streaming data processing system like the way windows are dealt with or performance issues in such a system. Computational complexity is abstracted from it and deals with properties inherent to algorithms.

Performance in practice can be characterized in many ways. Some performance metrics in the context of streaming data processing are: (average) processing time or latency per element of a stream as time per element, throughput or output rate as elements per unit of time, memory space usage of the system and power usage [19, 36]. The on-line processing of continuous and voluminous / unbounded streams by in-memory streaming data processing systems inherently has to deal with processing time and memory space usage constraints [10, 19, 36]. Thus performance metrics related to actual processing time and actual memory space usage are of primary interest and are a practical way to asses performance in practice. In the context of aggregates being periodically computed (upon every slide) over the retention, measuring the time it takes to compute the aggregate is effectively measuring output latency or processing time. From the average processing time the throughput can be determined. Measuring memory space usage of the system captures the memory used for elements of the stream and windows, synopsis structures and other system overhead. Power usage is not of interest to the stakeholder Info support.

The empirical part of the research in chapters 5, 6 and 7 elaborates on the following proposed steps to measure and model performance in practice:

1. Set environment (e.g. streaming data processing system).
2. Choose workloads representing output semantics (input, aggregate, accuracy requirements, retention and slide).
3. Choose approaches and synopsis structures.
4. Measure performance of approaches on workloads.

5. Model performance costs of each approach in terms of workload:
    - Processing time models.
    - Memory space usage models.

The configuration of the environment and measuring of actual processing time and memory space usage is is described in chapter 5. The environment is the machine, operating system, streaming data processing system and runtime. In the interest of Info Support, the streaming data processing system used is Microsoft StreamInsight [3, 4, 52, 58]. It fits in their stack of Microsoft products used by the Business Intelligence unit and fixed the environment. Processing time is measured as the actual time it takes to compute a workload under an approach (implementation) on the environment. The processing time can be divided into maintenance time of the synopsis structure and lookup/query time to compute the aggregate thereof. This mainly provides more detailed insight into the incremental approach. Such a distinction does not apply to the instantaneous approach. Memory space usage is measured as the actual amount of memory it takes for the streaming data processing system when computing an aggregate under an approach (implementation) on the environment. Lower granularity of memory space measurements are not feasible in .NET, in which the aggregates are implemented, and is obstructed by the closed source of the streaming data processing system StreamInsight.

The workloads represent the output semantics (input, aggregate, accuracy requirements, retention, slide). They determine *what* is to be computed and are approach and environment agnostic. The number of workloads is heuristically reduced in the interest of time. In the absence of non-synthetic data, a uniform random number generator is used to produce a stream of elements as input to the approaches. Only the selection of aggregates of (most) interest to Info Support and corresponding synopsis structures, discussed in chapter 3 and 4 respectively, are used during the empirical part of the research. The acceptable relative error of an approximation in a business case is likely to be no more than 5% (hence an accuracy level of 95%). This effectively limits the range of relative errors of interest. The retention and slide are chosen such that they affect measurement of processing time in the second to millisecond and memory space usage in the megabyte to kilobyte resolution. This should prevent minor fluctuations in performance from skewing the measurements. To obtain insight into both the effect of retention and slide (inversely: overlap) on performance, combinations of fixed slide with variable retention and variable slide with fixed retention are chosen. Varying retention demonstrates the effect of large window processing versus synopsis overhead, whereas varying slide demonstrates reprocessing versus synopsis overhead. Details of the heuristic reduction, as outlined above, is found in chapter 5 and summarized in table 5.4.

Regarding measurements, some care is required. It takes some time before the streaming data processing system reaches a steady state as the system needs to initialize and a full retention must be obtained (e.g. an hour worth of data). Measurements before the steady state can be discarded as they skew performance results. The performance of a workload and an approach is measured in isolation, i.e. no other workloads and / or approaches are executed on the environment and measured simultaneously. This prevents the correlation of performance with other approaches and workloads running simultaneously. Next to measuring quantitative performance by means of processing time and memory space usage, observed accuracy can be measured to indicate qualitative performance of approximations. It provides guidelines on approximations, e.g. how well accuracy requirements are met. This is of practical interest to Info Support. Note that to measure observed accuracy an exact and approximating approach need to have their aggregate output compared. Hence by an earlier argument on isolation, no quantitative performance should be measured at the same time.

The measurements make up the empirical results presented and discussed in chapter 6. To generalize the measurements on the environment, regression models are derived as described in chapter 7. The regression models can be plugged into the theoretical model of chapter 3 to compare the approaches empirically. Regression analysis is a widely known approach to model relationships from data [24, 42, 51, 55]. It is used to obtain a regression model for processing time and memory space usage per aggregate per accuracy requirements and per approach (during the research the input was fixed and hence of no interest). The regression models are parameterized on retention and slide, fitting nicely into the theoretical model of 3. The regression models are trained on part of the measurements. The other part is used for validating the error of the models independently. Model validation is a common practice [42, 59, 64]. This kind of validation is relatively simple and termed: holdout. Some common statistics, e.g. standard errors, the coefficient of determination regarding model fit, and plots (in the appendix B) are reported for the models. They are used to assess the (statistical) quality of the regression models. The environment setup, empirical results from the measurements and regression models generalizing the measurements on the environment answer the third subquestion in section 1.3.2.

1.4.3 Design Space

The means to determine how to compute an aggregate function over sliding windows is provided by the design space in chapter 8. Basically the design space describes how approaches can be compared regarding their performance. From a performance perspective, the best performing approach is the proposed way to compute the aggregate. As is clear from the beginning of this chapter, performance plays an important role in streaming data processing. At the heart of the design space lies the theoretical model of chapter 3. The performance of the approaches can be compared by scalability using computational complexity or empirically by using the regression models as empirical approximations of performance (in practice). Along the way guidelines have been derived: from the theoretical model on the computation of aggregates, from the properties of aggregates, from the synopsis structures and from the empirical results and regression analysis. Such guidelines augment the design space, i.e. the complement the performance comparisons and can generally be helpful to determine which approach is best to compute an aggregate.

The use of the theoretical model to compare the performance of the approaches and the guidelines answer the fourth subquestion in section 1.3.2. Moreover the main research question can be answered from the perspective of performance: the design space is a means to determine which approach is most performance cost effective in computing aggregate functions over sliding windows.

1.5 Structure of the Document

Formal semantics underlying streaming data processing and the approaches are defined in chapter 2. They should provide a common understanding next to informal descriptions earlier this chapter. A chapter on aggregates 3 follows, introducing the theoretical performance cost model that relates the computation of aggregates to the approaches in terms of performance. Properties of aggregates mentioned in literature are explored. The selection of aggregates of (most) interest to Info Support is made. The chapter on synopsis structures 4 relates synopsis structures to the incremental approach. Guided by the selected aggregates, a few synopsis structures along with examples are discussed. The environment setup for measuring performance in practice is treated in chapter 5. This is followed by a summarization and discussion of the empirical results from measurements in chapter 6. The measurements are generalized on the environment by regression analysis in chapter 7. The design space follows in chapter 8. It discusses the use of the theoretical model to compare the approaches in terms of performance. Finally conclusions are stated and future work is indicated in chapter 9.

Chapter 2

Semantics of Streaming Data Processing

Semantics for streaming data processing, often in the context of particular systems, are described in literature [6, 14, 46, 48, 52, 56, 58]. This chapter is inspired by the literature and aims to provide a concise formal definitions next to the informal descriptions in the introductory section on the approaches 1.2. Note that no attempt is made to be complete or to provide a universal formal model.

The discussions starts with a few notational conventions in section 2.1 used in the definitions. Next notions of time, streams and selections thereon are given in section 2.2. The discussion continues with several different windows being defined in section 2.3. The definitions for the approaches follow in section 2.4. Some constraints regarding the approaches are explicated in section 2.5. The chapter concludes with a wrap up in section 2.6.

2.1 Conventions

The following conventions are used to express the semantics in the definitions, inspired by the Z notation and functional languages:

- Function: an n-ary function typed $f : T_1 \times T_2 \times ... \times T_{n-1} \to T_n$ and applied as $f\, arg_1\, arg_2 ... arg_n = f(arg_1, arg_2, ..., arg_n)$ where arguments are typed $arg_1 : T_1, arg_2 : T_2, ..., arg_n : T_n$. If from the context the type of the function is clear, explicit typing is omitted.

- Finite sequence: a finite sequence over X is a total function $x : 0, 1, ..., n-1 \to X$ for some $n : \mathbb{N}$. In this case x is denoted by $\langle x_0, x_1, ..., x_{n-1} \rangle$ with $x_i = x\, i = x(i)$ and $\#x = n - m$. Furthermore $dom\, x = 0, 1, ..., n-1$ are the indices and $ran\, x = \{x_0, x_1, ..., x_{n-1}\}$ the set of elements.

- Infinite sequence: an infinite sequence over X is a total function $x : \mathbb{N} \to X$. In this case x is denoted by $\langle x_0, x_1, ... \rangle$ with $x_i = x\, i = x(i)$. Furthermore $dom\, x = \mathbb{N}$ are the indices and $ran\, x = \{x_0, x_1, ...\}$ the set of elements.

- Extraction: for $U \subseteq \mathbb{N}$ and sequence x an extraction is defined as $U \upharpoonleft x = \langle x\, u_0, x\, u_1, ... \rangle$ where $u_0, u_1, ...$ is the finite or infinite increasing enumeration of $\{i : dom\, x \mid i \in U\}$. For $k : \mathbb{N}$ a segment of sequence x is a sequence of the form $i ... i + k - 1 \upharpoonleft x = \langle x_i, ..., x_{i+k-1} \rangle$ (and it has domain 0 ... k - 1).

- Filter: for $U \subseteq \mathbb{N}$ and sequence x a filter is defined as $x \upharpoonright U = \langle x\, i_0, x\, i_1, ... \rangle$ where $i_0, i_1, ...$ is the finite or infinite increasing enumeration of $\{i : dom\, x \mid x_i \in U\}$.

- Map: define f^* over a sequence x as a mapping $f^*\, x = \langle f\, x_0, f\, x_1, ... \rangle$ for a unary function f.

2.2 Streams

In this section the notion of time, (logical) stream, time monotonic ordering of elements in a stream and (finite) selections of the stream by index and time are introduced. First a notion of a time domain, time instances and span of time units are introduced.

Definition 2.2.1 (Time)**.** *Let Time be a countably infinite set linearly ordered by $\leq$ and assume its members can be enumerated in increasing order $t_0, t_1, t_2, \ldots$. By t_0 is meant min(Time). The set Time is called the time domain and each member t_i is called a time instance. To simplify notation, it is assumed that $\Delta = t_j - t_i$ is interpreted as $t_{i+\Delta} = t_j$; Δ represents a span of units of the time domain.*

A stream is a (potentially) infinite sequence of objects. Objects have a payload, e.g. a temperature measurement, and a start and end time denoting its lifetime.

Definition 2.2.2 (Stream)**.** *Let Obj be a set of objects with $payload : Obj \rightarrow Payload$ and $start, end : Obj \rightarrow Time$ where $start\, o \leq end\, o$ for all objects $o : Obj$. Let an infinite sequence of such objects be a stream S denoted as type Stream.*

A short note on physical and logical streams is in place. In any practical system, the elements of a stream need not arrive in time order and lifetimes of elements can be change to some extent (e.g. end time, as long the element is not yet processed). Other issues could be thought of as well. However, the interest is not on capturing the messy details of elements at the physical level, but rather to abstract from them to a logical level. Hence implicitly it is assumed that only streams at the logical level are of interest. The details on how to deal with issues at the physical level is beyond the scope of this study. The following property guarantees elements in the (logical) stream are in time order.

Definition 2.2.3 (Time monotonicity)**.** *Elements in a stream S are monotonically increasing with Time, hence $start\, s_{i'} \leq start\, s_i$ whenever $i' \leq i$.*

To extract or select finite sequences of objects from the stream, e.g. used by definitions of windows discussed later, a notion of selection by index and time of objects from a stream is introduced.

Definition 2.2.4 (Selection)**.** *Elements in a stream S can be selected by index:*

- *For $i : \mathbb{N}$ let S_i be the element in S with index i.*
- *For $i : \mathbb{N}$ let sequence $S_{...i} = 0...i \upharpoonleft S$ or alternatively $S_{...i} = \langle s_0, ..., s_i \rangle$.*
- *For $i', i : \mathbb{N}$ let sequence $S_{i'..i} = i'...i \upharpoonleft S$ or alternatively $S_{i'...i} = \langle s_{i'}, ..., s_i \rangle$.*

Elements in a stream S can be selected by time:

- *For $t : Time$ let sequence $S_t = S \upharpoonright \{e : ran\, S \mid start\, e = t\}$; time is not unique.*
- *For $t : Time$ let sequence $S_{...t} = S \upharpoonright \{e : ran\, S \mid start\, e \leq t\}$.*
- *For $t', t : Time$ let $S_{t'...t} = S \upharpoonright \{e : ran\, S \mid t' \leq start\, e \leq t\} S$.*

2.3 Windows

This section introduces the, overloaded, notion of a window being a finite subsequence of a stream. Along with windows the extent and slide of a window, depending on the actual kind of window, are defined. First a global definition of a window is given.

Definition 2.3.1 (Window)**.** *Let a window with a start time, end time and segment of stream S be a data type $Window : Time \times Time \times Stream$. Let the following be defined for a window:*

- *$start\, (t_s\, t_e\, s) = t_s$ returns the start time of a window.*
- *$end\, (t_s\, t_e\, s) = t_e$ returns the end time of a window.*
- *$seq\, (t_s\, t_e\, s) = s$ gives the segment of stream S of a window.*
- *$minstart\, w = min\{e : ran\, seq\, w \bullet start\, e\}$.*
- *$maxstart\, w = max\{e : ran\, seq\, w \bullet start\, e\}$.*
- *$maxend\, w = max\{e : ran\, seq\, w \bullet end\, e\}$.*

Clearly the start and end time of the window bound the start and end times of the elements in the sequence, $start\, w \leq minstart\, w \wedge maxend\, w \leq end\, w$ for any window.

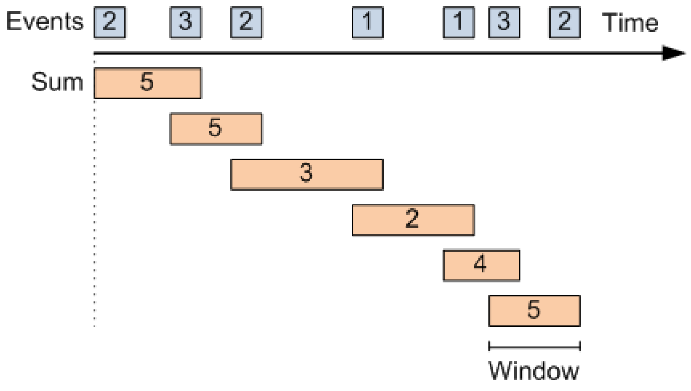


Figure 2.1: Visualization of a count-based window. Elements in the stream are above the time axis. Below the time axis the sliding windows are shown with the sum of the elements they contain. The windows have an extent of 2 and a slide of 1 element.○

Specific windows can be constructed depending on the way elements are selected for a window. The first kind of window is one in which exactly N elements are included. It is illustrated in figure 2.1.

Definition 2.3.2 (Count-based window)**.** *A count-based window w_c has a sequence which is a tail segment of stream S with exactly N positions, hence $seq\, w_c = i'...i \upharpoonright S \wedge \#(dom\; seq\, w_c) = N$. Assume $start\, w_c = minstart\; w_c$ and $end\, w_c = maxend\; w_c$. Let the following be defined for such a window:*

- *$extent\; w_c = \#(dom\; seq\; w_c) = N$.*
- *$slide\; w_c\; w'_c = |min\{e : dom\; seq\; w_c\} - min\{e : dom\; seq\; w'_c\}| = M$ elements.*

A slightly different kind of window is one in which elements with exactly N distinct start time are included. Clearly such windows can have $\geq N$ elements.

Definition 2.3.3 (Count-based distinct-time window)**.** *A count-based window with distinct start times w_d at a time instance t has a sequence with exactly N distinct start times of stream S, hence $seq\, w_d = S_{t'...t} \wedge \#\{e : ran\; seq\, w_d \bullet start\; e\} = N$. Assume $start\, w_c = minstart\; w_c$ and $end\, w_c = maxend\; w_c$. Let the following be defined for such a window:*

- *$extent\; w_d = \#\{e : ran\; seq\; w_d \bullet start\; e\} = N$.*
- *$slide\; w_d\; w'_d = \#\{e : ran\; (S \upharpoonright minstart\; w_d...minstart\; w'_d) \bullet start\; e\} = M$ distinct start times, assuming $minstart\; w_d \leq minstartw'_d$.*

A purely time based window is one in which elements are included belonging to a certain span of time with respect to some given time; elements within the time boundaries of the window. The actual number of elements such a window contains, depends on how many elements in the stream fall within the time boundaries. It is illustrated in figure 2.2.

Definition 2.3.4 (Time-based window)**.** *A time-based window w_t at time instance t and span of N time units has a sequence with elements of stream S whose start times are in $[t-N, t]$. Moreover $startw_t = t-N$ and $end\, w_t = t$ so that $seq\, w_t = S_{...t} \upharpoonright start\; w_t...end\; w_t$. Let the following be defined for such a window:*

- *$extent\; w_t = end\; w_t - start\; w_t = N$.*
- *$slide\; w_t\; w'_t = start\; w_t - start\; w'_t = M$ time units, assuming $start\; w_t \leq start\; w'_t$.*

A window whose extent equals its slide is said to be a tumbling window.

2.4 Approaches

This section provides the formal notions of batches and large windows. A notion of synopsis structure is introduced. Finally the approaches are defined formally, which were discussed informally in the section on the approaches 1.2.

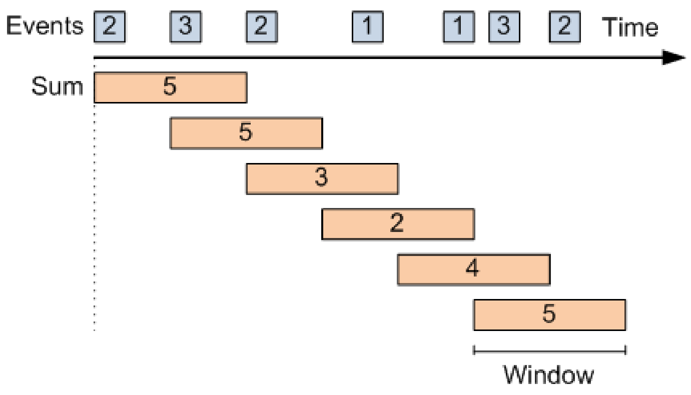


Figure 2.2: Visualization of a time-based window. Elements in the stream are above the time axis. Below the time axis the sliding windows are shown with the sum of the elements they contain. The windows have an extent of some time units and a slide of half that time units.∘

Definition 2.4.1 (Batches / small windows)**.** *An S partition is a sequence of windows $ws = \langle w_0, w_1, ...\rangle$ such that $start\ w_0 \leq start\ w_1 \leq ...$ and $seq\ w_i = S \upharpoonright (start\ w_i ... start\ w_{i+1} - 1)$. The sequences of the windows are adjacent and disjoint. The sequence ws is said to contain batches or small windows.*

Large windows are obtained by concatenating batches. They are defined next.

Definition 2.4.2 (Large windows)**.** *An S partitioning sequence composed from concatenating windows of ws is $WS = \langle W_0, W_1, ...\rangle$ so that there exist $start\, W_0 \leq start\, W_1 \leq ...$ and $end\, W_0 \leq end\, W_1 \leq ...$ with $seq\, W_i = \frown/\{w : ran\ ws \mid start\, W_i \leq start\ w \wedge end\ w \leq end\, W_i \bullet seq\ w\}$. Let the following be defined for such composed windows:*

- *Expired part: $W_i - W_{i+1}$ is $\frown/\{w : ran\ ws \mid start\, W_i \leq start\ w \leq end\ w \leq start\, W_{i+1} - 1 \bullet seq\ w\}$.*
- *New part: $W_{i+1} - W_i$ is $\frown/\{w : ran\ ws \mid end\, W_i + 1 \leq start\ w \leq end\ w \leq end\, W_{i+1} \bullet seq\ w\}$.*

The former is the part removed at the beginning of W_i and the latter appended to W_i to obtain W_{i+1}. The sequence WS is said to contain large windows composed of batches / small windows.

Putting everything together, the instantaneous and incremental approaches can be defined.

Definition 2.4.3 (Approaches)**.** *The (abstract) goal is to compute some (aggregate) function f over each $W_i : WS$ by the following approaches:*

- *Instantaneous: compute $f\ W_i$, or alternatively $f^*\ WS = \langle f\ W_0, f\ W_1, ...\rangle$.*
- *Incremental: an auxiliary function g working on sequences such that, for adjacent large windows W_i and W_{i+1}, the value $g\, W_{i+1}$ to approximate $f\ W_{i+1}$ can be computed from:*
  - *the part $W_i - W_{i+1}$*
  - *the part $W_{i+1} - W_i$*
  - *and the value $g\, W_i$.*

  In this case, the auxiliary function g resembles the synopsis structure over a large window. The synopsis over W_{i+1} by the value $g\, W_{i+1}$ reflects the new elements by the batches / small windows in $W_{i+1} - W_i$ and accounts for the expired elements in $W_i - W_{i+1}$ with respect to the synopsis over W_i by the value $g\, W_i$.

This research is about: (aggregate) functions f computed or approximated in what way (approaches with synopsis structures and algorithms) and at what performance costs (processing time and memory space usage).

2.5 Constraints

Note that in many streaming data processing systems [3, 4, 18, 21, 22, 53], including StreamInsight used for this research, windows have a constant extent and slide. In other words, given a window specification (e.g. for a query) each window processed has the same constant extent and adjacent windows have the same constant slide. This applies to the large windows of the instantaneous approach and batches of the incremental approach. Adjacent batches have a constant slide equal to their extent so they are consecutive and disjoint. It is assumed empty windows are not of interest in the processing. In the worst case, there is no overlap between consecutive retentions (there is no interest in slides larger than the retention). This leads to the following definitions.

Definition 2.5.1 (Retention constraint)**.** *The constant number of elements $R \geq 1$ to aggregate over is the retention; an approach-agnostic term. For large windows of the instantaneous approach it should hold that: $\forall W \in WS \mid \#ran\, W = R$. The synopsis structure of the incremental approach $g\, W_i$ should cover the retention.*

Definition 2.5.2 (Slide constraint)**.** *The constant number of elements S is the slide, so that $1 \leq S \leq R$. For any adjacent large windows of the instantaneous approach W_i, W_{i+1} it should hold that $\#ran\,(W_{i+1} - W_i) = \#ran\,(W_i - W_{i+1}) = S$. The synopsis structure of the incremental approach $g\, W_{i+1}$ should accurately reflect the slide with respect to $g\, W_i$.*

Definition 2.5.3 (Batch / small window constraint)**.** *The constant number of elements B that are in a batch / small window, so that $1 \leq B \leq \gcd(R, S)$ and $B \mid R \wedge B \mid S$. For the batches / small windows in the incremental approach it should hold that $\forall w \in ws \mid \#ran\, w = B$ and for any adjacent batches / small windows their slide is implied to be B (as they are defined to be consecutive and disjoint).*

Note that for count-based large windows and batches / small windows the constraints should not be a problem. For count-based distinct-time and time-based large windows and batches / small windows an estimation is required, because the presence of duplicate elements in a stream and the variable number of elements with a timespan cause a variable number of elements in such a window. In that case R, S and B should be determined from the average or maximum rate of elements in a stream (in practice).

2.6 Wrap-up

This chapter provides formal definitions on the semantics of common terminology in streaming data processing, used throughout this document and during this research. Important is that elements (object) in the stream are time monotonic and as such one operates on a logical notion of a stream; messy details of physical streams like dealing with out-of-order data need not be dealt with. The streaming data processing system discussed in a later chapter 5 has implementations of operators necessarily deal with windows on logical stream. Three kinds of windows are distinguished: count-based, count-based distinct-time and time-based windows. Count-based windows contain exactly N elements, count-based distinct-time windows contain at least N elements and time-based windows contain as many elements as its time interval contains in the stream.

The approaches are defined in terms of batches / small windows, synopsis structures to be maintained and queried and concatenations of batches into large windows over which (aggregate) functions are computed. Synopsis structures themselves are discussed in chapter 4. Some practical constraints are imposed acceptable in the context of actual streaming data processing systems. The retention R is the number of elements aggregated over under either approach, the slide S is the number of elements by which the retention slides as new elements arrive under either approach and B is the number of elements in a batch / small window implying the slide of the batches too. For the interested reader, the literature references in the introduction of this chapter provide pointers to more and different definitions, e.g. other kinds of windows beyond the scope of this research, and system specific definitions.

Chapter 3

Aggregates

This chapter derives a theoretical performance cost model of processing time and memory space usage. It relates the computation of aggregates to the approaches in terms of performance in an abstract way. The model is discussed in section 3.1. It can be used to assess scalability by instantiating it with computational complexity as discussed in section 3.2. By scalability it is argued in some cases approximation is the only way to reduce memory space usage. Moreover (small) batches are justified in case of large slides. A quick reference to instantiating the theoretical model with empirical approximations is given in 3.3. In chapters 5, 6 and 7 the way to obtain those approximations is fleshed out.

In section 3.4 of this chapter properties of aggregates mentioned in literature are discussed and their scalability related to the approaches. The chapter ends with a selection of aggregates of interest to Info Support to guide the discussion on synopsis structures in chapter 4 and for the empirical part of the research in chapters 5, 6 and 7 in the interest of time. The instantaneous computation of the aggregates is discussed shortly. The chapter ends with a wrap-up in section 3.6.

3.1 Theoretical Model

Aggregates are particular functions that compute some value over a bunch of values. In the context of streaming data processing, an aggregate function f is computed over a retention of R elements of the stream and as new elements arrive in the stream, the retention will slide by S elements. Conceptually the retention behaves as a sliding window. The approaches determine how the retention is materialized: a large window in the instantaneous approach and a synopsis structure updated by batches / small windows of B elements in the incremental approach. For simplicity it is assumed that R, S, and B are known and constrained as described in chapter 2.

In general, let the processing time of computing an aggregate function f over a large window of R elements by the instantaneous approach be denoted by $T(R)$. Potentially the stream is very large/has infinite length, so the average processing time costs to process n retentions can be expressed as:

$$\lim_{n \to \infty} \frac{n * T(R)}{n} = T(R)$$

Hence the average processing time costs converge to the costs of computing the algorithm over the R elements in a large window. This is not a surprise as each large window is processed independently. The processing time costs are independent of the slide. In general, let the memory space usage of computing an aggregate over a large window of R elements by the instantaneous approach be denoted by $S(R)$. Note that memory space can be reused, unlike processing time.

In the incremental approach batches of B elements are processed. There are $\frac{S}{B}$ batches in the slide and $\frac{R}{B}$ batches in the retention. The first $\frac{R}{B}$ batches are used to construct the first retention. Upon a slide of $\frac{S}{B}$ batches the second retention can be constructed. The next slide causes the third retention to be constructed and so on. This requires the following number of batches to process n retentions:

$$\frac{R + (n-1) * S}{B}$$

Each element in the batch updates the synopsis, called update or maintenance time, denoted by $T_u(R)$. This represents the processing time to update a synopsis over the last R elements for a new element and includes taking care of expiration. It is the update time per element! Additionally as the number of batches processed equals a slide, the synopsis structure is used to look up or query the value for the aggregate function f over the retention of R elements. This is called the update or query time and denoted by $T_q(R)$; the time to query the synopsis over R elements. It is the time per lookup or query! The processing time costs over n retentions then is:

$$\frac{R + (n-1) * S}{B} * B * T_u(R) + n * T_q(R)$$

Potentially the stream is very large/has infinite length, so the average processing time costs to process n retentions can be expressed as:

$$T(R, S, B) = \lim_{n\to\infty} \frac{\frac{R + (n-1) * S}{B} * B * T_u(R) + n * T_q(R)}{n}$$

$$= \lim_{n\to\infty} (\frac{R}{n} + \frac{S * n}{n} - \frac{S}{n}) * T_u(R) + T_q(R) = S * T_u(R) + T_q(R)$$

Hence the average processing time costs converge to the update time of the synopsis for a slide of S elements and a query on the synopsis structure. This is not a surprise as for every slide, the S new elements arrived are used to update the synopsis over R elements and once every slide the synopsis over R elements is queried. Interestingly this is independent of the batches. Moreover the costs of the initial retention of $\frac{R}{B}$ batches can be neglected, assuming the number of processed retentions n is sufficiently large. The space of a batch of B elements and a synopsis structure over R elements, called maintenance space, is denoted by $S(B, R) \leq S(S, B)$ (since by definition $1 \leq B \leq \gcd(S, R) \leq S$). Again note that memory space can be reused, unlike processing time.

3.2 Scalability Model

The theoretical model can be used to compare the approaches in terms of computational complexity, hence scalability. The theoretical model for the approaches should have $T(\cdot)$ and $S(\cdot)$ instantiated with computational complexity. The definitions [8] below can be used to compare the approaches by substituting the instantiated theoretical model for the functions in the definitions below, e.g. $T(R)$ as $f(x)$. An example is discussed later, see 3.2.5.

Definition 3.2.1. *A function $f \in O(g)$, i.e. f grows no faster than g, if* $\lim_{x\to\infty} \dfrac{f(x)}{g(x)} = c < \infty$ *exists, including $c = 0$.*

It means $f(x) \leq c * g(x)$ for some real constant $c > 0$ and nonnegative integer $x \geq x_0$. In such a case f is bounded above by g. The function f does grow faster than g, if the limit is ∞.

Definition 3.2.2. *A function $f \in \Omega(g)$, i.e. f grows at least as fast as g, if* $\lim_{x\to\infty} \dfrac{f(x)}{g(x)} = c > 0$ *exists, including $c = \infty$.*

It means $f(x) \geq c * g(x)$ for some real constant $c > 0$ and nonnegative integer $x \geq x_0$. In such a case f is bounded below by g.

Definition 3.2.3. *A function $f \in \Theta(g)$, i.e. f grows at the same rate as g, if* $\lim_{x\to\infty} \dfrac{f(x)}{g(x)} = c$ *and $0 < c < \infty$.*

It means f is in order g. The set $\Theta(g)$ is the set of functions in both $O(g)$ and $\Omega(g)$, i.e. $\Theta(g) = O(g) \cap \Omega(g)$. Effectively $\Theta(g)$ determines an equivalence class or complexity class.

The instantaneous approach keeps at least the entire retention of R elements in memory as a large window and aggregates over those elements, regardless of the aggregate. This leads to the following proposition.

Proposition 3.2.4. *In the instantaneous approach processing time and memory space usage are both in $\Omega(R)$.*

This does not hold for the incremental approach. By the choice of $R > 1$ (non-trivial aggregation), S and B, properties of the incremental approach can be identified independent of the aggregate and synopsis structure.

Suppose the slide is minimized so that overlap is maximized, hence S to 1 :

$$\lim_{S \to 1} T(R,S,B) = T_u(R) + T_q(R)$$

$$\lim_{B \to 1} S(B,R) = S(1,R) \text{ as } 1 \leq B \leq S$$

Paradoxically, sliding one element at a time is not necessarily the best case regarding performance of the incremental approach. To see why, suppose the following problem [27, 28]: a on a stream of 0's and 1's, maintain the count of 1's over the last R elements. To maintain an exact count, one requires $\Theta(R)$ bits of memory. Suppose the number of 1's over the last R elements is m. Upon a new element, an old element expires. The new element can be inspected and increase m if it is a 1. The old element is either a 0 or a 1, potentially decreasing m. If the old element was a 1 and not stored, one would have no way to maintain an exact count.

This is not the only aggregate causing trouble. Suppose a stream of elements whose values arrive in descending order (e.g. 10, 9, 8, 7, ...). Maintaining the exact maximum over the last R elements requires space $\Omega(R)$ [35]. Upon sliding one element, the first element of the last R elements is the new maximum, hence requiring at least all last R elements to be stored. A similar argument holds for the minimum. Maintaining exact variance over the last R elements requires space $\Omega(R)$ as well [65]. The bottom line seems to be that maintaining exact aggregates over the last R elements sliding every new element has space requirements linear in R. Approximation of the aggregates by synopsis structures reduces space requirements at the cost of accuracy [5, 9, 27, 35]. It is discussed in the chapter on synopsis structures 4. For exact computation in the incremental approach, the most trivial synopsis seems to be justified, one that maintains the last R elements (of interest). Such a synopsis for the incremental approach has no space benefit over the instantaneous approach, however existence of sub-linear update time $T_u(R)$ and query time $T_q(R)$ has processing time benefits. An example demonstrates it.

Example 3.2.5. Suppose one wants to compute the sum over R elements in the retention sliding every $S = 1$ element. For the instantaneous approach it requires processing time and memory space usage in $\Omega(R)$, hence from a computational complexity point of view $T(R) = R$ and $S(R) = R$. For the incremental approach using the trivial synopsis structure R elements need to by stored, e.g. in a circular buffer, and an internally maintained sum incremented for new elements and decremented for expired elements in the synopsis. Clearly it takes $O(2)$ to update the internally maintained sum for each new element and a corresponding expired element. It takes $O(1)$ to insert the new element into the buffer representing the synopsis structure (overwriting the oldest element). Moreover it takes $O(1)$ to query / look up the internally maintained sum. So update time $T_u(R) = 2 + 1 = 3$ and query time $T_q(R) = 1$. Using definition 3.2.1:

$$\lim_{R \to \infty} \frac{T(R,S,B)}{T(R)} = \lim_{R \to \infty} \frac{S * 3 + 1}{R} = \lim_{R \to \infty} \frac{3+1}{R} = 0$$

Hence $T(R,S,B) \in O(T(R)$ and by reversing the ratio $T(R) \in \Omega(T(R,S,B))$ by definition 3.2.2 and $T(R) \notin \Theta(T(R,S,B))$ by definition 3.2.3. Clearly the instantaneous approach bounds processing time from above regarding the incremental approach with a trivial synopsis. By definition 3.2.3 it can be shown that $S(B,R) \in \Theta(S(R))$, hence memory space usage grows at the same rate. $\diamond$

Now suppose the slide S is maximized so that overlap is minimized and batch size B is either maximized or minimized, hence S to R and B to 1 or B to $S = R$:

$$\lim_{S \to R} T(R,S,B) = R * T_u(R) + T_q(R) \in \Omega(R)$$

$$\lim_{B \to 1} S(B,R) = S(1,R) \text{ or } \lim_{B \to S} S(B,R) = S(R,R) \in \Omega(R) \text{ for } S \text{ to } R$$

Clearly processing time of the incremental approach becomes similar to the instantaneous approach, hence at least linear in the number of elements in the retention. Besides, the reuse of the overlap between consecutive retentions by the synopsis structure loses its benefit. Note that this case shows the memory space usage benefits of small sized batches, i.e. it justifies batches that are smaller than the slide. This is obvious from the limits on the batch size B. A limit of B to 1 shows the synopsis determines the memory space usage. A limit of B to S, which by transitivity is R, requires space at least linear in the number of elements in the retention R.

3.3 Empirical Model

The interesting part of the theoretical models is the genericity of the time and space functions $T(\cdot)$ and $S(\cdot)$. In the previous section they were instantiated by functions of computational complexity used to show and compare the scalability of the approaches. However, the time and space functions could equally well be replaced by empirically established approximations. Suppose one measures the processing time and memory space usage for various output semantics (consisting of input, aggregate, retention, slide and accuracy requirements). From the measurements one derives empirical models capturing performance trends, e.g. processing time of an aggregate under the instantaneous approach for various combinations of retention and slide. Such models can be used to approximate performance and hence approximate the time and space functions $T(\cdot)$ and $S(\cdot)$.

Such measurements and empirical models provide insight into the performance in practice. This can show when one approach outperforms another, e.g.:

- In case both approaches belong to the same complexity class. As complexity is asymptotic, other terms and multiplicative factors are hidden. Actual performance can show which one performs better in practice (i.e. concrete streaming data processing system and approach implementation).
- In case unexpected side-effects occur, e.g. a bug or performance issue in the streaming data processing system affecting performance in practice.

The empirical part of the research is discussed in later chapters; see empirical results in chapter 6, empirical approximations by regression models derived from the empirical results in chapter 7 and their use in the design space in chapter 8.

3.4 Properties of Aggregates

Up until now the characteristics of the aggregates regarding their computation have not been discussed, but were abstracted from by $T(\cdot)$ and $S(\cdot)$. In literature properties of aggregates have been identified [42] and reused to determine computational characteristics of aggregates and use the properties in techniques to compute aggregates more efficiently in the context of sliding windows [7, 47], but also in sensor networks [50] and multi-dimensional data cubes [38]. Those properties are touched upon shortly and reused by some synopsis structures discussed in chapter 4. In table 3.1 some well-known aggregates mentioned in literature have their properties indicated. No attempt is made to be complete. Some examples follow in section 3.4.3. The interested reader can use literature references for an in-depth discussion and particular applications.

3.4.1 Boundedness and Differentiality

The discussion is started with a notion of boundedness of aggregates. Boundedness refers to the a priori bounds of the state requirements.

Definition 3.4.1 (Boundedness / State requirements)**.** *An aggregate f over a dataset X can be computed from a sub aggregate function g over disjoint partitions $X_1, X_2, X_3, ..., X_n$ where $\bigcup_{1\leq i\leq n}(X_i) = X$ and super aggregate function h in $f(X) = h(\{g(X_i)|1 \leq i \leq n\})$. Boundedness is the extent to which g bounds the state requirements in computation of the sub aggregates.*

The boundedness of aggregates divides aggregates into bounded and unbounded ones.

Definition 3.4.2 (Bounded aggregate)**.** *A bounded or non-holistic aggregate has constant size state requirements to summarize a partition, i.e. the partial state for a sub aggregate $g(X_i)$ is bounded. The following bounded aggregates are distinguished:*

- *Distributive: the partial state of the sub aggregates $g(X_i)$ is of the same size as the state of the super aggregate h and $f = g$.*
- *Algebraic: the super aggregate can be computed from summary or synopsis functions $g(X_i)$ of the partitions and can be stored in constant size state. The super aggregate is of different size than the sub aggregates, hence $f \neq g$. In this case h super aggregates over the synopses.*

Aggregate	**Bounded**	**Unbounded**	**Differential**	**Duplicate Sensitive**	**Robust**	**Monotonic**
Average	x [38, 47, 50]		x	x [50]	x [50]	[50]
Center of Mass	x [38]		x	x	x	
Count	x [38, 47]		x [50]	x [50]	x	x [50]
Count Distinct		x [38]			x [50]	x [50]
Covariance	x		x	x	x	
Histogram		x [50]	?	x [50]	x [50]	? not in [50]
Heavy Hitters/Frequent Items		x [38, 47]	x [7]	x	x	x
Linear Regression	x		?	x	x	?
Kurtosis	x		x	x	x	
Max	x [38, 47, 50]				[50]	x [50]
MaxN	x [38]			x		x
Median		x [38, 50]		x [50]	[50]	[50]
Min	x [38, 47, 50]				[50]	x [50]
MinN	x [38]			x		x
Mode/Most Frequent		x [38]		x		
Quantile		x		x		
Rank		x [38]		x		
Skewness	x		x	x	x	
Sum	x [38, 47, 50]		x [7]	x [50]	x	x [50]
Variance (Std. Deviation)	x [38]		x	x	x	x

Table 3.1: Overview of some well-known aggregates from literature and their properties. Aggregates for which a property holds, are marked (and vice versa). Question marks indicate uncertainty. ∘

Definition 3.4.3 (Unbounded aggregate)**.** *An unbounded or holistic aggregate has no constant size state requirements to summarize a partition, i.e. the partial state for a sub aggregate $g(X_i)$ is unbounded and proportional to the size of its partition. Two special cases of holistic aggregates are defined:*

- *Unique: the amount of partial state for a sub aggregate $g(X_i)$ is proportional to the number of unique values in its partition.*
- *Content-sensitive: the amount of partial state for a sub aggregate $g(X_i)$ is proportional to some, e.g. statistical, property of the values in its partition.*

The following introduces the notion of a differential aggregate, sometimes referred to as being invertible [25].

Definition 3.4.4 (Differential / Subtractable)**.** *An aggregate is said to be differential if for any partition $Y \supseteq X$ there is $f(Y - X) = h(\{g(Y), g(X)\})$ and $f(Y) = j(\{g(Y - X), g(X)\})$ where $|g(X)| < |X|$ (for $|X| > 1$). The following notions of a differential aggregate are distinguished:*

- *Full-differential: the partial state of the sub aggregates $g(X_i)$ is of constant size; bounded aggregates that are differential.*
- *Pseudo-differential: the partial state of the sub aggregates $g(X_i)$ has no constant size; unbounded aggregates that are differential.*

Intuitively for differential aggregates h is subtractive or decremental and j is additive or incremental. Note that g can be a synopsis. The condition $|g(X)| < |X|$ prevents $g(X)$ from being the identity function and rather summarize X. An example of a full-differential aggregate is the average and an example of a pseudo-differential aggregate is a fixed-width or equi-width histogram. Some incremental (and decremental) functions for aggregates like sum, count, average, variance (also standard deviation), skewness and kurtosis have been identified in literature [12, 20].

The properties of boundedness and differentiality relate to exact computation of aggregates by the incremental approach, as the approach is inherently partitioned by batches / small windows of B elements. This does not apply to the instantaneous approach. A bounded aggregate has a sub aggregate in constant space (hence a priori bounded). An unbounded aggregate has the sub aggregate in non-constant space, proportional to the size of the batch, leaving little room for useful sub aggregation. The retention is partitioned into $\frac{R}{B}$ batches, each with a sub aggregate. Note that the bounded aggregate can maintain an aggregate over the entire retention internally in constant space.

Proposition 3.4.5. *Memory space usage of a bounded aggregate is at least $(\frac{R}{B} + 1) \in \Omega(R)$ to store the constant size sub aggregates over the batches and a constant size internally maintained aggregate over the retention.*

Proposition 3.4.6. *Memory space usage of an unbounded aggregate is at least $\frac{R}{B} * B = R \in \Omega(R)$ to store the non-constant size sub aggregates over the batches (each sub aggregate proportional to B).*

The choice of S and B does not affect space asymptotically when the batches are used as a partitioning. It is easy to see that in practice, if B is large with respect to R, a bounded aggregate is likely to require far less storage than an unbounded aggregate. Obviously, if B is small then storage is likely to become similar. Clearly as B goes to 1, sub-linear space requirements force one into the synopsis structures approximating the aggregate. Some are discussed in the chapter 4 on synopsis structures, were section 4.4 exploits boundedness. The linear memory space usage requirements provide a justification of approximations reducing memory space usage at the cost of accuracy.

For a bounded differential aggregate the batch is aggregated by using the incremental function B times, combining individual elements of the batch into a sub aggregate. Additionally, each new sub aggregate is applied to an internally maintained aggregate using the incremental function. This is done for $\frac{S}{B}$ batches to cover a slide. Each new sub aggregate is stored. Moreover the sub aggregates of the $\frac{S}{B}$ expired batches need to be undone from the internally maintained aggregate using the decremental function and can be removed thereafter. The incremental and decremental functions operate in constant time $O(1)$ as they operate on constant size sub aggregates and a constant size aggregate maintained internally. As sub aggregates can be maintained in time order of the batches by a linked list (e.g. from oldest sub aggregate to most recent sub aggregate), appending a new sub aggregate or removing the oldest takes constant time $O(1)$.

Proposition 3.4.7. *Processing time to update a bounded differential aggregate internally is* $\frac{S}{B} * (B + 1 + 1) + \frac{S}{B} * (1 + 1)$ *per slide, yielding amortized time per element in the slide* $(1 + \frac{4}{B}) \sim O(1)$; $T_u(R)$. *The internally maintained aggregate can be queried / looked up in constant time* $O(1)$; $T_q(R)$.

A non-differential aggregate has no such incremental and decremental functions. Sub aggregating B elements in a batch takes time $\Omega(B)$ and is repeated $\frac{S}{B}$ times to cover a slide. As no aggregate is maintained internally, $\frac{S}{B}$ expired batches can be removed immediately. Again a linked list can be maintained in time order, where an append or removal takes constant time $O(1)$. The aggregate over the retention is determined by processing at least the $\frac{R}{B}$ sub aggregates.

Proposition 3.4.8. *Processing time to update a bounded non-differential aggregate takes at least* $\frac{S}{B} * (B + 1) + \frac{S}{B}$ *per slide, yielding amortized time per element in the slide* $(1 + \frac{2}{B}) \sim \Omega(1)$; $T_u(R)$. *To query / look up the aggregate at least all sub aggregates need to be reprocessed once requiring time* $\frac{R}{B} \sim \Omega(R)$; $T_q(R)$.

An unbounded differential aggregate can maintain the aggregate internally. Storing the internally maintained aggregate requires space linear in the number of elements in the retention, next to storing the non-constant size sub aggregates. A batch is aggregated by using the incremental function B times. Each sub aggregate is used to update an internally maintained aggregate. This is done for $\frac{S}{B}$ batches to cover a slide. Each sub aggregate is stored. Moreover the sub aggregates of the $\frac{S}{B}$ expired batches need to be undone from the internally maintained aggregate and can be removed thereafter. However, unlike bounded aggregates, the sub aggregates of unbounded aggregates are proportional to B. Hence one can only be sure the incremental and decremental functions take time $\Omega(1)$. The $\frac{S}{B}$ sub aggregates can be removed thereafter. Again sub aggregates are maintained in time order in a linked list.

Proposition 3.4.9. *Processing time to update an unbounded differential aggregate takes at least* $\frac{S}{B} * (B + 1 + 1) + \frac{S}{B} * (1 + 1)$ *per slide, yielding amortized time per element in the slide at least* $(1 + \frac{4}{B}) \sim \Omega(1)$; $T_u(R)$. *The internally maintained aggregate can be looked-up in constant time* $O(1)$; $T_q(R)$.

An unbounded non-differential aggregate has no such incremental and decremental functions. Unlike the bounded aggregates, the sub aggregates of unbounded aggregates are proportional to B. The use of sub aggregates is questionable in this case. Sub aggregating a batch takes time $\Omega(B)$ plus the overhead of storing the sub aggregate, whereas simply storing the elements of a batch takes $O(B)$. It is performed for the $\frac{S}{B}$ batches in a slide. Removing a sub aggregate of an expired batch happens in constant time $O(1)$, whereas removing the elements of an expired batch takes $O(B)$. This is performed for $\frac{S}{B}$ expired batches. After storing the elements, the aggregate over the retention is determined in $\Omega(R)$. In case of sub aggregates it takes $\Omega(\frac{R}{B})$, where an additional multiplicative factor of B might be justified as sub aggregates are proportional in B. Effectively, storing elements instead of sub aggregates, resembles the trivial synopsis that stores all elements in the retention. The benefit is an upper bound on the update time.

Proposition 3.4.10. *Processing time to update an unbounded non-differential aggregate takes* $\frac{S}{B} * B + \frac{S}{B} * B$ *per slide, yielding amortized time per element at least* $(1+1) \sim O(1)$ *per element;* $T_u(R)$. *To query / look up the aggregate the retention need to be reprocessed requiring time* $\Omega(R)$; $T_q(R)$.

Note the use of $\Omega(\cdot)$ and $O(\cdot)$ in the propositions. For the bounded differential aggregates the upper bounds are known. For the others, they are not obvious without knowledge of the aggregate and corresponding algorithm. For example, non-differential aggregates require at least all sub aggregates to be processed to determine the aggregate over the retention (and perhaps more). Furthermore, unbounded aggregates have sub aggregates proportional in B that complicate the determination of the bounds. Tighter bounds can be obtained by looking at algorithms (and synopsis structures) to compute actual aggregates.

3.4.2 Duplicate Sensitivity, Robustness and Monotonicity

In literature [50] a few other properties of aggregates are mentioned that are interesting, but hard to fit into the approaches or theoretical model. They are discussed for completeness of the discussion on properties of aggregates.

Definition 3.4.11 (Duplicate sensitivity). *The sensitivity to duplicate elements over which is aggregated, i.e. whether or not an aggregate is affected by duplicate elements. Mathematically an operation is duplicate if it is idempotent [25].*

Handling duplicates requires additional techniques for detecting and filtering them. Regarding the semantics and correctness of the output, it should be dealt with. Techniques to do so are beyond the scope of this research.

Definition 3.4.12 (Robustness / Tolerance of loss). *An exemplary aggregate returns one or more representative elements, while a summary aggregate computes a value over all elements over which is aggregated. The latter is more robust or tolerant to loss of elements, whereas the former is not.*

Loss can be unintentional, e.g. in case of packet loss in a network, or intentional, e.g. sampling. The former is beyond the scope of this research. The latter clearly stems from using approximation techniques. It seems to boil down to the robustness of the approximation. Clearly if robustness needs to be accounted for, strict error bounds on the approximations are required. Techniques must be adapted for or be suitable to such bounds. Approximations by synopsis structures are discussed in chapter 4.

Definition 3.4.13 (Monotonicity). *Suppose X_1 and X_2 partition the dataset of elements Y, an aggregate f is said to be monotonic if it holds that $\forall X_1, X_2 : f(Y) \geq max(f(X_1), f(X_2)) \vee f(Y) \leq min(f(X_1), f(X_2))$. In other words the super aggregate $f(Y)$ bounds the sub aggregates $f(X_i)$.*

Monotonicity can be helpful in pushing down predicate evaluation in a distributed setting to lower communication costs [50]. It is an issue beyond the scope of this research. It seems of little use in the non-concurrent and non-distributed context of this research.

3.4.3 Examples of Properties of Aggregates

Some aggregates in table 3.1 are discussed in examples below to provide an intuition for the properties of boundedness, differentiality, duplicate sensitivity, robustness and monotonicity.

Example 3.4.14. From table 3.1 it can be seen that the variance is a bounded aggregate that is (full) differential, duplicate sensitive, robust and monotonic. It is bounded because the variance over partitions is represented by the constant size tuple $\langle count, average, variance\rangle$ (and even $\langle count, sum, variance\rangle$ as average is defined by a sum and a count). The following incremental functions [12, 20], for which its decremental counter parts can be derived easily, show the differentiality:

$$count(X_{i,j}) = count(X_i) + count(X_j) \tag{3.1}$$

$$average(X_{i,j}) = \frac{count(X_i) * average(X_i) + count(X_j) * average(X_j)}{count(X_{i,j})} \tag{3.2}$$

$$variance(X_{i,j}) = variance(X_i) + variance(X_j) + \frac{count(X_i) * count(X_j)}{count(X_i) + count(X_j)} * (average(X_i) - average(X_j))^2 \tag{3.3}$$

For other aggregates such functions exist too, like skewness and kurtosis. The presence of duplicates affects the variance, potentially shifting it. However it is robust as it provides a summary over all the data, rather than being a selection of some particular elements. The variance is monotonic as is obvious from equation 3.3, where the variance of $X_{i,j}$ is always greater than or equal to the variance of its partitions X_i and X_j. ◇

Example 3.4.15. From table 3.1 quantiles belong to the unbounded aggregates that are non-differential, duplicate sensitive, exemplary and non-monotonic. A quantile of an ordered dataset X is a value x such that the fraction of values in the data less than or equal to x is ϕ [17]. Put differently, the quantile is the value x whose rank in ordered dataset of size N is $\lceil \phi * N \rceil$ [5]. The median is the quantile with $\phi = 0.5$, hence the middle of the dataset. Duplicates are part of the ordered dataset and have rank too, hence they affect the quantiles. Since values with particular ranks (e.g. median) are returned, the aggregate is exemplary. Monotonicity does not hold as the distribution of X can be different from its partitions X_i and X_j (which themselves can be different). Hence in general the quantiles of X are not obvious from the quantiles of X_i and X_j. ◇

3.5 Selection of Aggregates

Using the theoretical model, one can compare the approaches. However, up until now no synopsis structures other than the most trivial one, storing all elements, are discussed. Obviously the incremental approach has more to offer by plugging in other synopsis structures. The computational complexity of the synopsis structures (update and query / lookup time) can be plugged into the scalability model of section 3.2 for a more comprehensive comparison. From the wealth of aggregates and synopsis structures, a selection must be made in the interest of time and to guide the discussion on synopsis structures. Moreover, as indicated in section 3.3 empirical measurements can provide insight into performance of the approaches in practice and be plugged into the empirical model to approximate performance. In the interest of time a limited number of aggregates can be assessed empirically. To this end two aggregates of (most) interest to Info Support are selected which will be used to guide the discussion on synopsis structures in chapter 4 and the empirical part of the research in chapters 5, 6 and 7.

The business intelligence unit of Info Support considered the following aggregates of main interest: average, count, distinct count, maximum, minimum, quantiles, sum, top-k and variance / standard deviation. These are common aggregates found frequently in the software they use. From this shortlist, the aggregates variance and quantiles are chosen. Variance being the square of standard deviation, is a well-known and widely used measure of spread. It is a bounded differential aggregate and as such is a representative of aggregates like average, count, kurtosis, skewness and sum (see table 3.1) computed in more or less the same way. As such it provides nice coverage of some aggregates of interest to Info Support. It has seemingly beneficial properties regarding processing time (see proposition 3.4.7) and memory space usage (see proposition 3.4.5) in the incremental approach as opposed to the instantaneous approach for exact computation (see proposition 3.2.4). In the instantaneous approach variance can be computed by using the incremental function in example 3.4.14 in a single pass over the large window of R elements.

Proposition 3.5.1. *In the instantaneous approach it takes processing time $O(R)$ and memory space $O(R)$ to compute variance.*

Quantiles describe the distribution of the data. A quantile ϕ of an ordered dataset of length N is a value with rank $\lceil \phi * N \rceil$ in the dataset such that all values before it are no larger. The median is a particular quantile with $\phi = 0.5$, minimum the quantile with $\phi = 0.0$ and maximum the quantile with $\phi = 1.0$. Quantiles belong to the unbounded non-differential aggregates and as such provide challenges to processing time (see proposition 3.4.10) and memory space usage (see proposition 3.4.6) in the incremental approach as opposed to the instantaneous approach for exact computation (see proposition 3.2.4). Approximation by synopsis structures seems interesting to improve on memory space usage in general and possible processing time. As sub aggregates are proportional to B, sub aggregation seems of little use. However, unless using the trivial synopsis structure, approximation is required to reduce memory space usage. In the instantaneous approach quantiles can be computed by using a balanced binary search tree (e.g. a red-black tree) in which all elements of the large window of R elements are kept in value order. Adding elements to that tree takes $O(\log R)$. A single pass over the tree yields the quantiles of interest. The tree is stored alongside the large window, both containing the same elements.

Proposition 3.5.2. *In the instantaneous approach it takes processing time $O(R*\log R+R) \sim O(R*\log R)$ and memory space $O(2*R) \sim O(R)$ to compute quantiles.*

3.6 Wrap-up

Theoretical performance cost models for the instantaneous and incremental approach are defined. It relates the computation of aggregates to the approaches in terms of performance costs. For the instantaneous approach the average processing time is $T(R) \in \Omega(R)$, the time to compute the aggregate over the elements in the retention. It has a memory space usage of $S(R) \in \Omega(R)$, determined by the space to compute the aggregate over R elements and keep the R elements in the retention. For the incremental approach the average processing time is $T(R, S, B) = S * T_u(R) + T_q(R)$, the time to update the synopsis for S element in the slide and a query / lookup on the synopsis structure to obtain the aggregate. It has a memory space usage of $S(B, R)$, determined by the space to keep the batch of B elements and a synopsis structure over R elements in the retention.

Substituting $T(\cdot)$ and $S(\cdot)$ with empirical models from measurements, performance in practice can be compared. The empirical part is discussed in chapters 5, 6 and 7. Substituting $T(\cdot)$ and $S(\cdot)$ by computational complexity, the approaches can be compared regarding scalability. The scalability model is used to demonstrate the linear space requirements of exact computation of aggregates in case of sliding 1 element (maximizing overlap). It justifies approximation to reduce memory space usage. On the other hand the scalability model is used to justify (small) batches especially when the slide is large (minimizing overlap).

In exact computation boundedness of aggregates can reduce memory space usage significantly (in practice) when batches contain several elements. Moreover differentiality can reduce update and query / lookup time, hence processing time. Boundedness, differentiality and other properties mentioned in literature for some well-known aggregates are summarized in table 3.1. Bounded differential aggregates have a constant time upper bound on update and query / lookup time. This cannot be stated about bounded non-differential and unbounded aggregates (regardless of differentiality). Tighter bounds can be obtained by looking at algorithms for aggregates.

For a more comprehensive comparison using the models, synopsis structures need to be explored. To guide the discussion, aggregates of interest to Info Support are chosen. They are used during empirical part of the research again. The selected aggregates are variance and quantiles. The former a bounded differential aggregate, the latter an unbounded non-differential aggregate. Their instantaneous computation is discussed shortly.

The following **guidelines** regarding the computation of aggregates are derived from this chapter:

- The instantaneous approach requires processing time and memory space usage in $\Omega(R)$.
- Exact computation of aggregates with a slide of 1 element has memory space usage $\Omega(R)$, justifying approximations trading accuracy for reduced memory space usage; processing time depends on the aggregate.
- Exact computation of aggregates can exploit boundedness to reduce memory space usage and differentiality to reduce processing time, in particular when the batches a not singletons.
- Duplicate sensitive aggregates require additional techniques to filter duplicates, regardless of the approach taken.
- Non-robust aggregates seem more susceptible to loss of e.g. sampling and (perhaps) require more robust approximation (e.g. strict error bounds).

Chapter 4

Synopsis Structures

This chapter describes how synopsis structures relate to the incremental approach. It elaborates on a few selected synopsis structures that can be used in the incremental approach. Their computational complexity can be plugged into the theoretical model of chapter 3 regarding scalability. Moreover the synopsis structures are used by the incremental approach during the empirical part of the research in chapters 5, 6 and 7. Additionally they help shape the design space in 8.

The discussion is started with desiderata of synopsis structures mentioned in literature in section 4.1. In section 4.2 these are extended with requirements imposed by the incremental approach, leading to a taxonomy of synopsis structures. In the interest of Info Support a few conceptually different synopsis structures are explored that cover the taxonomy, rather than a particular one in-depth with all its optimizations, to provide a broad view on the subject. The synopsis structures are explored in the context of the selected aggregates in chapter 3: variance and quantiles. Some generally applicable synopsis structures are discussed with examples of their workings in sections 4.3, 4.4 and 4.5. Aggregate-tailored synopsis structures follow in section 4.6 for variance and in section 4.7 for quantiles. The chapter ends with a wrap-up in section 4.8.

4.1 Desiderata from Literature

Synopsis structures are developed to deal with large volumes of data in a space and perhaps time efficient way; a small surrogate for a dataset expensive or impossible to access [33] (in whole). Such structures are useful in e.g. approximate query and join estimation, computing aggregates and data mining [2]. Regarding the incremental approach, the synopsis structure is used to reflect the retention and overlap upon sliding in a space and time efficient way (or one at the expense of the other). In contrast, the instantaneous approach operates on the retention by a large window without a synopsis structure.

In literature some desiderata of synopsis structures are mentioned [2, 6, 10, 33, 36]:

- One-pass constraint: synopsis structures must be maintained in a single pass over the data.
- Performance: maintenance of the synopsis structures and computing the aggregate from it should require as little resources, processing time and memory space usage, as possible. This is expressed in literature as computational time and space complexity for the following:
  - Update time: the processing time to update the synopsis for a new element of data. It relates to $T_u(R)$ in the theoretical model of chapter 3.
  - Query / Lookup time: the processing time to compute the aggregate from the synopsis. It relates to $T_q(R)$ in the theoretical model of chapter 3.
  - Maintenance space: the memory space usage of the synopsis. It relates to a part of $S(B, R)$ in the theoretical model of chapter 3.
- Robustness: the degree to which a synopsis structure is accurate; conformance to an application specific error metric or bound. It seems to relate to the notion of robustness of aggregates in definition 3.4.12 of chapter 3.
- Applicability: synopsis structures with narrow applicability require a different synopsis structure per application (i.e. aggregate). A broadly applicable synopsis structure can be used to compute

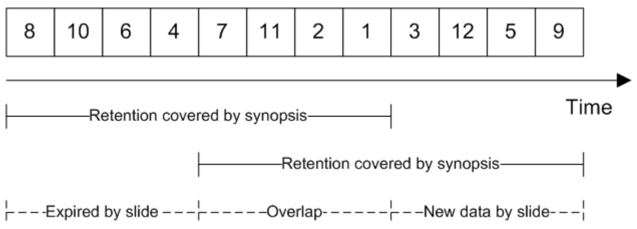


Figure 4.1: Elements of the stream are shown above the time axis. The retention is covered by the synopsis structure. As it slides, old elements expire implicitly and new elements are presented explicitly. The overlapping part need not be reprocessed. ○

several aggregates and potentially allow the synopsis structure to be shared across windows or operators.

- Evolution sensitive: as the distribution of data in the stream unlikely is stable over time, synopsis structures should account for such evolution of the stream.

The one-pass constraint and performance desiderata are common in streaming data processing, as discussed shortly in the introductory chapter 1. The one-pass constraint holds for all synopses explored in this research, as in the incremental approach the retention is not readily available like in large windows of the instantaneous approach. Robustness is important when aggregates are approximated using synopsis structures in the incremental approach. Synopsis structures being accurate most of the time (globally) but providing high errors occasionally (locally) are not considered robust. Referring back to the motivational case in the introductory chapter 1, a helium level of an MRI scanner that is way off occasionally can be very costly and perhaps even dangerous. Maintenance of the maximum helium level must be accurate at all times. Ideally an error bound or metric is adhered to by a synopsis structure, e.g. an error of at most 1% is guaranteed. Applicability and evolution sensitivity are of a lesser concern. Significant memory space savings or substantially increased processing time can justify the loss of applicability. The evolution sensitivity is an issue if synopsis structures capture an entire stream, however by definition the retention bounds the data to the recent past. Evolution is not considered of interest in this research.

4.2 Taxonomy

The incremental approach cannot handle arbitrary synopsis structures, even if they meet the desiderata from literature, but have specific requirements. Most important, a synopsis structure in the incremental approach must support a notion of retention. In other words, the synopsis structure must cover the most recent yet bounded past reflecting the retention and thus be applicable to sliding windows. Note that the retention is a particular way of decaying elements. Conceptually a weight function assigns a weight to elements in the stream between 0.0 and 1.0 [23]. A weight of 0.0 indicates no contribution (expired), 1.0 full contribution and everything in between some less extreme contribution. The elements in the retention have a weight of 1.0, while all other elements have a weight of 0.0. Alternatives are exponential and polynomial (time-)decay, though they are beyond the scope of this research. Several synopsis structures can be found in literature like sampling, hashing, counting, histograms, wavelets and sketches [2, 10, 36]. However, many of them build a synopsis over the entire past, i.e. the entire stream [2, 28].

A hidden assumption for synopsis structures over a retention is: elements are implicitly deleted [5]. Whereas new elements are explicitly presented to a synopsis structure, expired elements are not and so they are implicitly deleted. Consequently a synopsis structure over the retention must handle those expired elements itself. By the assumption it is not needed to store the entire retention (to be able to perform explicit deletes)! Potentially this comes at the expense of accuracy as will be clear from several synopsis structures discussed later. The synopsis structure and retention with explicit new elements and implicit deletes of expired elements is shown in figure 4.1.

A subtle issue with the retention is whether it contains a fixed or variable number of elements and so resembles the behavior of fixed-size or variable-size sliding windows. In chapter 2 count-based windows in definition 2.3.2 contains a fixed number of elements. Count-based distinct-time windows in definition 2.3.3 and time-based widows in definition 2.3.4 contain a variable number of elements. In literature [5, 11] this distinction leads to different or adapted synopsis structures with a different computational complexity (e.g. update time, query time and maintenance space of a synopsis structure). Note that a fixed-size is a special case of a variable-size, nevertheless the distinction is important.

The extent to which the retention is reflected by the synopsis structure, affects the accuracy of the aggregates obtained from it and hence the conformance with the accuracy requirements of the output semantics. Effectively this means the synopsis structures must be robust, as stated by the desiderata in the previous section. In literature [5, 9, 27, 39, 40, 65] various synopsis structures approximating aggregates are defined that adhere to a relative error (upper) bound $\epsilon \in [0, 1]$ or an error metric. Moreover a success probability (lower) bound $1 - \delta \in [0, 1]$ of achieving the relative error bound can be involved [5]. Obviously, synopsis structures producing exact aggregates are robust and have a relative error $\epsilon = 0$ and success probability $1 - \delta = 1$. Synopsis structures producing approximate aggregates are only of interest if they can achieve a relative error $\epsilon < 1$ with success probability $1 - \delta > 0$. The relative error bound and success probability are of interest to the stakeholder Info Support, because it provides a level of confidence on the aggregates. Moreover it can guide their decision on which synopsis structure to use in the context of a business case, e.g. constraints on accuracy. The accuracy requirements for the selected aggregates variance and quantiles in chapter 3 are defined as follows, based on literature [5, 9, 39, 49]

Definition 4.2.1 (ϵ-approximate variance)**.** *For exact variance V and approximate variance $\hat{V}$, the ϵ-approximate variance is:*

$$\hat{V} \ in \ (1 \pm \epsilon) * V \ hence \ \frac{|V - \hat{V}|}{V} \leq \epsilon$$

In a randomized / probablistic context it is:

$$P\left(\frac{|V - \hat{V}|}{V} \leq \epsilon\right) \geq 1 - \delta$$

Definition 4.2.2 (ϵ-approximate quantiles)**.** *Suppose an ordered dataset of size R. For a ϕ-quantile with rank $r = \lceil \phi * R \rceil$ and rank $\hat{r}$ of the approximated quantile, the ϵ-approximate ϕ-quantile is:*

$$\hat{r} \ in \ \lceil (\phi \pm \epsilon) * R \rceil = \lceil \phi * R \pm \epsilon * R \rceil = \lceil r \pm \epsilon * R \rceil \ hence \ \frac{|r - \hat{r}|}{R} \leq \epsilon$$

In a randomized / probablistic context it is:

$$P\left(\frac{|r - \hat{r}|}{R} \leq \epsilon\right) \geq 1 - \delta$$

Putting the requirements together, the following taxonomy of synopsis structures applicable to the incremental approach is proposed:

- Fixed number of elements (in the retention):
  - Exact ($\epsilon = 0$)
  - Approximate($\epsilon > 0$):
    - Deterministic ($1 - \delta = 1$)
    - Randomized / probablistic ($1 - \delta < 1$)
- Variable number of elements (in the retention):
  - Idem.

The remainder of this chapter will discuss synopsis structures applicable to the selected aggregates of (most) interest to Info Support discussed in chapter 3. The synopsis structures will cover the taxonomy as to provide Info Support a broad view on the matter, rather than an in-depth treatment of a particular synopsis structure and its optimizations. Moreover the discussion is started with some generally applicable synopsis structures followed by aggregate-tailored synopsis structures.

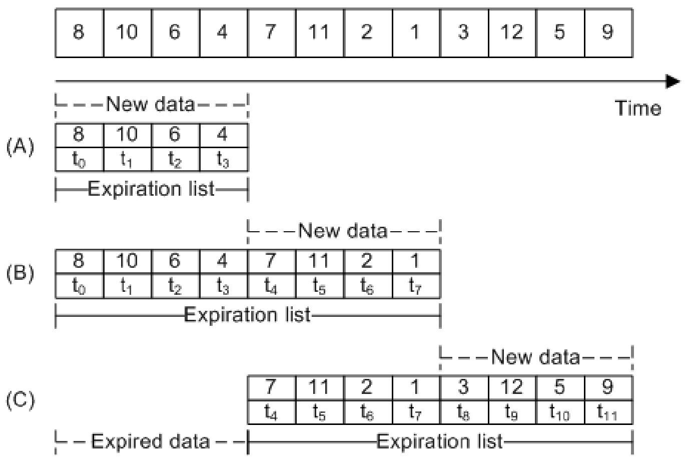


Figure 4.2: The initial batch of new data is stored in the expiration list making up the retention (a). The second batch of data is stored too (b). After the third window is processed the oldest batch of elements are removed and the elements of the third batch inserted (c). ◦

4.3 Expiration List

The expiration list is a simple way to support retention and implicit deletes in a synopsis structure. It basically stores every new element as a bucket with the payload of the element and its timestamp in a list and removes all buckets belonging to expired elements. It effectively maintains all R elements in the retention; a trivial synopsis. On a fixed-size retention a round-robin buffer (or ring/circular buffer) based on an array of length R can be used, keeping the elements in time-order. Note that timestamps need not be stored then, only the payloads of the elements. On variable-size retention a linked list can be used to remove old buckets in front and append new buckets to maintain time-order. Timestamps are required, e.g. to determine time-based expired elements. Those data structures can append (with overwrite) and remove in front in constant time. Similar to the instantaneous approach, all data to aggregate over is available. The expiration list is shown in figure 4.2 and illustrated in example 4.3.1.

Computing variance is straightforward. As it is bounded (see definition 3.4.2) and differential (see definition 3.4.4) an internally maintained variance can be stored in constant space and updated in by incremental and decremental functions [12, 20] in constant time. Every expired bucket undoes the payload from the internally maintained variance by a decremental function, while every new bucket applies the payload by an incremental function. As the retention slides, the overlap is not reprocessed when computing the aggregate. In chapter 6 the performance in practice is shown.

Computing quantiles is more difficult. Quantiles are determined by ranks over an ordered dataset. This is possible by using a (balanced) binary search tree in logarithmic time. A sequential scan over the tree is required to determine the quantiles. Some options are available:

- Keep the buckets only in time-order and upon a query sort and scan the buckets to obtain quantiles.
- Keep the buckets only in value-order (instead of time-order), requiring a scan to determine the expired buckets and another scan to obtain the quantiles.
- Keep the buckets both in time-order and value-order, requiring a scan to obtain the quantiles.

The first has the fastest update time and the slowest query time. It seems most appropriate on large slides, effectively amortizing the cost of querying. The second has the slowest update and query time. The third has the largest space requirement, but the fastest query time. It seems most appropriate on small slides, though at the expense of memory space usage. The third option is chosen, as it has good update time and the fastest query time. Memory space usage is a limited concern for expiration lists anyway.

Example 4.3.1. Suppose the situation depicted in 4.2. Let one be interested in the average over the last 8 elements (retention count), sliding every 4 elements. The stream starts with the first batch, which can be inserted into the expiration list. The average over the expiration list is $28/4 = 7$. The second window updates the expiration list with 4 new buckets. The retention now covers 8 elements. The average is $49/8 = 6.125$. As the third window is processed the retention now reflects the contents of the second and third batch, hence the elements of the first batch of elements has expired. The expiration list is updated by removing the oldest 4 elements representing the implicit delete. The elements of the third batch are inserted. The expiration list covers the retention that slided 4 elements. The average now becomes $50/8 = 6.25$. $\diamond$

In summary, the expiration list is a conceptually simple synopsis to compute exact aggregates. Appending to a round-robin buffer or linked list takes time $O(1)$, removing from a linked list takes time $O(1)$. Adding to and removing from a (balanced) binary search tree takes time $O(\log R)$. For variance and bounded aggregates in general the internally maintained aggregate takes space $O(1)$ and can be queried in $O(1)$. A bounded differential aggregate has incremental and decremental functions taking time $O(1)$ per element to apply to or undo from an internally maintained bounded differential aggregate. Hence variance under an expiration list takes:

- Maintenance space $O(R + 1) \sim O(R)$, to keep the list and internally maintained variance.
- Update time $O(1 + 1 + 1 + 1) \sim O(1)$, to remove and undo an expired element and append and apply a new element.
- Query time $O(1)$, to lookup the internally maintained variance.

For quantiles (the third option) an expiration list takes:

- Maintenance space $O(2 * R) \sim O(R)$, to keep both the list and (balanced) binary search tree.
- Update time $O(1 + \log R + 1 + \log R) \sim O(\log R)$, to remove an expired element and append a new element to both the list and (balanced) binary search tree.
- Query time $O(R)$, to sequentially scan the (balanced) binary search tree for the quantiles.

The expiration list is a broadly applicable and robust synopsis structure that can compute various aggregates in an exact fashion. Furthermore it can be easily adapted to support fixed-size and variable-size retentions. It has no memory space usage benefit over the instantaneous approach. Note that variance is a bounded differential aggregate, hence the low update and query time. From the propositions on boundedness and differentiality of aggregates in chapter 3, it is clear that bounded non-differential and unbounded aggregates might have a different computational complexity.

4.4 Panes

With regard to the memory space usage, a simple improvement over expiration lists can be obtained by using the property of boundedness 3.4.1 of an aggregate as discussed in chapter 3. It is inspired by an optimization in streaming data processing called panes [47]. In the introduction chapter 1 it was touched upon as part of the discussion on alternatives to the incremental approach. The propositions in the discussion on the properties of aggregates in chapter 3 are related to it. Conceptually the retention is divided in two levels: the pane level and the retention level. At the pane level consecutive disjoint partitions are sub aggregated. At the retention level the sub aggregates are super aggregated into the final aggregate over the retention. If the retention contains R and the slide contains S elements, then at least $\frac{R}{\gcd(R,S)}$ panes are required. Similarly if the retention spans R_T and the slide spans S_T time units, then at least $\frac{R_T}{\gcd(R_T,S_T)}$ panes are required. Note how the number of panes coincides with the number of maximally sized batches under the constraints in chapter 3 on semantics. A straightforward approach is to use the batches / small windows as partitions.

Each partition is sub aggregated and represented as a pane. A new pane is appended to a list and an expired pane is removed. Again time-order is maintained. Similar to expiration lists, for a fixed-size retention a round-robin buffer can be used to append new panes and overwrite expired panes. For a variable-size retention a linked list can be used to append new panes and remove expired panes in the front. To support time-based retentions, each pane must store the timestamp of the oldest element in the partition it represents, next to the sub aggregate. The memory space usage reduction is obtained from partitions being sufficiently large and sub aggregates being sufficiently small. The latter is achieved by bounded aggregates (see definition 3.4.2). The benefit seems lost for unbounded aggregates whose sub aggregates are proportional in size to the partitions (see definition 3.4.3). Note that the property of differentiality of aggregates (see definition 3.4.4) can be used to lower processing time for bounded aggregates. For bounded aggregates an internally maintained aggregate (the super aggregate) can be

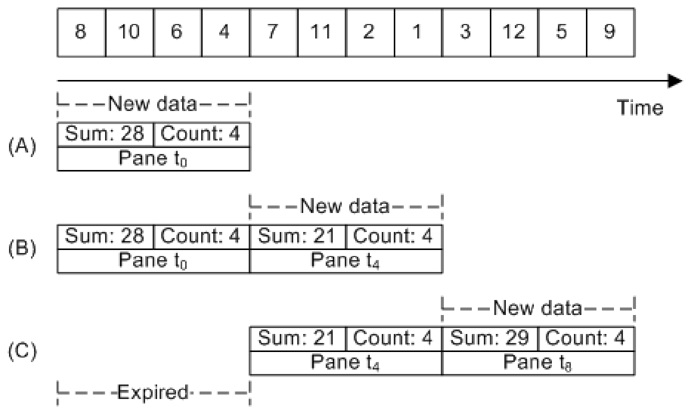


Figure 4.3: Each batch is sub aggregated and stored as a pane. The initial batch is covered by a pane (a). The second batch is covered by another pane (b), the panes can be super aggregated. The third batch is covered by yet another pane (c) and has the first pane expire. ∘

stored in constant space. By differentiality batches can be sub aggregated in constant time per element and sub aggregates be applied to / undone from the internally maintained aggregate in constant time. Regarding the selected aggregates variance and quantiles, only the former seems to benefit from the use of panes. Panes are depicted in figure 4.3 and illustrated in example 4.4.1.

Computing variance is straightforward. Suppose batches are used as partitions and an internally maintained variance (the super aggregate over the retention) is kept. Variance is computed over the batch (yielding a sub aggregate) using an incremental function [12, 20] that takes constant time per element. The variance over the batch is stored in the list (round robin buffer / linked list) in constant time. It is applied to the internally maintained variance using the incremental function. An expired pane is removed from the list in constant time. It is undone from the internally maintained variance using a decremental function in constant time.

Example 4.4.1. Suppose there is a stream of elements as in figure 4.3 and one is interested in the average over the retention of the last 8 elements, sliding every 4 elements. The number of panes making up the retention is $8/\gcd(8,4) = 2$. To determine the expiration of panes, each pane maintains the timestamp of the oldest element of the partition it covers. To compute the average, each pane additionally has to keep a sum and a count. Each batch turns into a pane. The first batch starts at timestamp t_0 has a sum of 20 and a count of 4; the average is $28/4 = 7$. The second batch becomes a pane with timestamp t_4, sum 21 and count 4. The two consecutive panes now cover the retention and can be super aggregated. The average is the sum of sums divided by the sum of counts over those panes, $(28 + 21)/(4 + 4) = 49/8 = 6.125$. The third batch at timestamp t_8 has a sum of 29 and a count of 4. The second and third pane cover the retention upon the sliding of 4 elements and cause the first pane to expire. The average now becomes $(21 + 29)/(4 + 4) = 50/8 = 6.25$. Note that keeping 2 panes of 3 pieces of data each, 8 elements of the stream were covered. Imagine space savings if the number of elements covered by the panes increases. ⋄

In summary, panes allow a conceptually simple synopsis structure that can provide great memory space savings in practice (not asymptotically), while still computing exact aggregates. Note that if the number of panes gets large (toward the size of the retention) or an unbounded aggregate is involved, the performance in practice will be similar to an expiration list. It seems to favor bounded aggregates, in particular if they are differential. Appending panes to a round-robin buffer or linked list takes time $O(1)$, removing a pane from a linked list takes time $O(1)$. For variance and bounded aggregates in general the internally maintained aggregate takes space $O(1)$ and can be queried in $O(1)$. A bounded differential aggregate has incremental and decremental functions taking time $O(1)$. It is required B times to sub aggregate a batch and required once to apply or undo a sub aggregate from an internally maintained bounded differential aggregate. Hence variance under a panes synopsis structure with batches as partitions takes:

- Maintenance space $O(\frac{R}{B} + 1) \sim O(R)$, to keep the buffer / list of panes and internally maintained variance. Note how it tightens the bound with respect to proposition 3.4.5 on memory space usage of bounded aggregates under batch partitioning.
- Update time $O(B + 1 + 1 + 1 + 1)$ per batch and amortized $O(1 + \frac{4}{B}) \sim O(1)$ per element in a batch. It is the cost to sub aggregate, apply to the internally maintained aggregate, store the pane,

undo the expired pane from the internally maintained aggregate and remove the pane. Note its resemblance with proposition 3.4.7 regarding update time.

- Query time $O(1)$, to lookup the internally maintained variance. Note its resemblance with proposition 3.4.7 regarding query time.

Panes provide a broadly applicable and robust synopsis structure that can compute various aggregates in an exact fashion. Furthermore it can be easily adapted to support fixed-size and variable-size retentions.

4.5 Sampling

Sampling is used to represent a population by some subset, the sample, which is constructed by including elements of the population into the sample with some probability [2]. In general one wants a small sample that represents the population well. A synopsis structure representing a sample is used often in a streaming data context [15]. The smaller the sample the more memory space savings, however potentially at the expense of accuracy. Such sample sizes can be bounded or unbounded, i.e. they have a bound on the size. Moreover sampling can be done with replacement, i.e. each element has the same probability of being included into the sample and can be included multiple times. Independence among sampled elements is assumed. For example if the population consists of the elements 1, 2, 3 and 4 each element can be included in the sample with probability $\frac{1}{4}$. Furthermore suppose after 1 is included in the sample by chance, sampling the next element could again include 1 by chance but 2, 3, and 4 equally likely. Alternatively sampling can be done without replacement, i.e. an element is included in the sample at most once. For example the initial probability of being included is $\frac{1}{4}$, however after 1 is included in the sample by chance only 2, 3 and 4 are left for sampling. From those three elements sampling the next one has a probability of $\frac{1}{3}$ for each of them. Sampling without replacement generally is more informative, but sampling with replacement is sometimes preferred due to its simplicity [15, 29]. A sampling technique is uniform if samples of equal size are produced equally likely [16]. More formally the probability mass function $\mathcal{P}(S; D)$, for a population D of elements and sample $S \subseteq D$ of elements, provides the probability of obtaining a sample S by a sampling technique applied to D. If for any D it holds that $\mathcal{P}(S; D) = \mathcal{P}(S'; D)$ where $S, S' \subseteq D$ and $|S| = |S'|$, then the sampling technique is said to be uniform. Some techniques to obtain a sample are discussed in the subsections.

In a sense, aggregating over a sample is similar to aggregating over an expiration list. Variance can be maintained internally and updated by new elements in the sample and old elements removed from the sample. Note that in sampling, one deals with sample variance rather than population variance! No literature was found on ϵ-approximate variance using sampling, hence it seems sampling cannot guarantee a bounded error with a given success probability. The sample must be sorted to determine the quantiles. Hence analogous to expiration lists a (balanced) binary search tree is maintained next to the sample. Interestingly, in literature [5] a sample size of $O(\epsilon^{-2} * \log(\epsilon * \delta)^{-1})$ is claimed to suffice for ϵ-approximate ϕ-quantiles with a success probability of $1 - \delta$.

4.5.1 Bernoulli Sampling

A straightforward unbounded uniform sampling technique without replacement is Bernoulli sampling [16, 30, 31]. Every element has a probability of $q \in [0, 1]$ of being included in the sample and consequently a probability of $1 - q$ of being excluded. Elements are independently included (or excluded) of each other. The probability mass function $\mathcal{P}(S; D) = q^{|S|}(1 - q)^{|D|-|S|}$ of this technique clearly is uniform. The main disadvantage is the unbounded sample size. Its expected size is $q|D|$. The sample size is binomially distributed with the probability of the sample size being k: $P(|S| = k) = \binom{|D|}{k} q^k (1 - q)^{|D|-k}$. Note that with small probability the sample size $|S| = |D|$. Bernoulli samples are easy to manipulate and have some interesting properties. If S_1 and S_2 are Bernoulli samples of D_1 and D_2 given that $D_1 \cap D_2 = \emptyset$, then $S_1 \cup S_2$ is a Bernoulli sample of $D_1 \cup D_2$. Furthermore if S is Bernoulli sample of D with q and S' is a Bernoulli sample of S with q', then S' is a Bernoulli sample of D with $q'q$. Repeated sampling until a sample of some bounded size is produced, is uniform but approximately Bernoulli at best [16]. Besides, it could be very expensive computationally. Another interesting property is that removing an element e from a Bernoulli sample S is simply the removal of e from S if it is in the sample, $S' = S \backslash \{e\}$.

In the context of the incremental approach with batches of B elements, one can take B Bernoulli samples (a sample per element). This yields a sample size $\leq B$ as empty samples can safely be ignored. Each sample has a timestamp used to determine when the sample expires and can be removed. By the properties of Bernoulli samples, the union of all non-expired samples yields a sample over the retention.

This resembles the stratified Bernoulli technique [16]. Moreover it is a sampled version of the expiration list. A panes-like variant would keep a sample over a partition in a pane and the timestamp of the oldest element. The sample for the retention is the union of the samples in the non-expired panes covering the retention.

4.5.2 Priority Sampling

A concrete uniform sampling technique using Bernoulli sampling, keeps a backing sample where each new element is inserted into with a certain probability [11, 15]. Expired elements are removed from the backing sample and a sample of size k is produced by down-sampling from the backing sample. This technique can be improved upon by eliminating the need for a backing sample and down-sampling. For the sequence-based sampling technique, suppose each element has an index i, e.g. 1, 2, 3, ... For a retention of R elements, a single element sample contains the i-th element with probability $\frac{1}{min(i,R)}$. If the i-th element is chosen, it also selects the element j that will replace the i-th element if it expires. Element j is the element chosen uniformly at random from the indexes $[i+1, i+R]$. If element j arrives, it chooses its successor element. Consequently a chain of replacements is created. Taking k such single element samples produces a fixed size uniform sample with replacement. The sequence-based sampling technique fails for a variable-size retention. A simple solution to this problem is priority sampling [11]. Each element in the retention is assigned a randomly chosen priority between 0 and 1. Only elements for which there is no element with both a later (non-expired) timestamp and a higher priority are stored. The element with the highest priority is reported as a sample. Again one can take k such single element samples. This yields a uniform sample with replacement.

A variant of the technique is called (uniform) bounded space priority sampling [29]. Starting with single element sampling, assume two elements are kept (at most): a candidate element from the current retention and a test element from the previous retention (before sliding). The technique works as follows:

1. If the current test element is double expired, i.e. has a timestamp before the previous retention, discard the test element.
2. If the current candidate element is expired, it becomes the test element (only the timestamp and priority are required).
3. As the i-th element arrives, generate a priority for it:
   (a) If there is currently no candidate element or the priority of the i-th element is larger than the priority of the current candidate element, the old candidate element is replaced by the i-th element.
   (b) Otherwise ignore the i-th element.
   (c) The sample reported is the candidate element if it is set and its priority is larger than the priority of the test element (or no such test element is set).

The extension to k sized samples without replacement requires the following modifications, where S_t is the set of test elements and S_c the set of candidate elements:

1. If there are double expired test elements, discard them from S_t.
2. If there are expired candidate elements, remove them from S_c and add them to S_t.
3. Keep at most k highest priority candidates S_c and at most k highest priority test elements S_t.
4. As the i-th element generate a priority for it:
   (a) If $|S_c| < k$ insert the i-the element.
   (b) Else if the i-th element has a priority higher than the element with the lowest priority in S_c, insert the i-th element and remove the element with the lowest priority.
5. The sample reported is $top-k(S_c \cup S_t) \cap S_c$ with $top-k(\cdot)$ denoting the k highest priority elements.

Bounded space priority sampling is shown in figure 4.4 and illustrated in example 4.5.1.

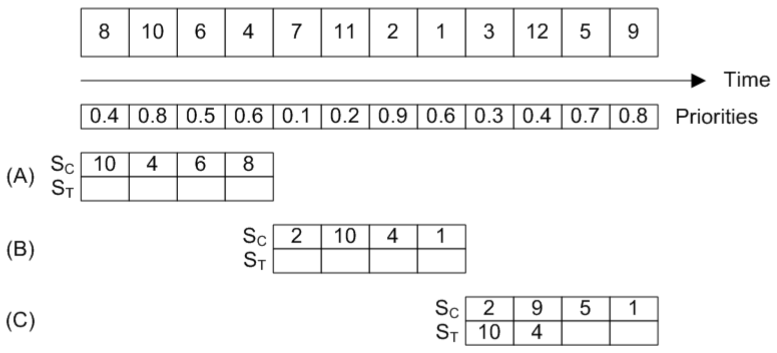


Figure 4.4: A $k = 4$ bounded priority space sample is maintained. Each element is assigned a random priority. The initial batch of data has all its elements become candidates in S_c and has no test elements S_t (a). The second batch has element 2 and 1 replace 6 and 8 (b), all of which are reported as the sample. Upon the third batch, the candidates 10 and 4 from the initial batch expire and become test elements (c). The highest priority elements of the third batch become candidates. The elements 2, 5 and 9 are reported as the sample of $k = 4$. ∘

Example 4.5.1. Suppose there is a stream of elements as in figure 4.4 and one is interested in the average over a retention of the last 8 elements, sliding every 4 elements. The average is approximated by a sample of size $k = 4$. Each element is assigned a random priority in $[0.0, 1.0]$ upon its arrival. Initially the set of candidate and test elements, S_c and S_t respectively, are empty. The first batch of data has all its elements become candidates (ordered by priority in 4.4). The reported sample is exactly the average over the batch and hence the average turns out to be exact, $28/4 = 7$. The second batch has element 2 replace 8 and element 1 replace 6 in the candidates. Now the sample is constructed over 8 elements covering the retention. The sample reported is $top-4(\{2, 10, 4, 1\} \cup \emptyset) \cap \{2, 10, 4, 1\} = \{2, 10, 4, 1\}$ yielding an average of $17/4 = 4.25$. Upon processing the third batch, the candidates 10 and 4 expire and become test elements, causing $S_c < k$. The elements 5 and 9 from the third batch become candidates. The sample reported is $top-4(\{2, 9, 5, 1\} \cup \{10.4\}) \cap \{2, 9, 5, 1\} = \{2, 9, 5\}$ yielding an average of $16/3 = 5\frac{1}{3}$. Upon processing a subsequent batch, the current test elements double expire and are removed. At the same time the candidate elements 2 and 1 expire and become test elements. The process continues. ◇

In summary, a backing sample being down sampled is a wasteful technique in particular regarding memory space usage. Sequence-based sampling improves upon that by creating a chain of successors. It has expected maintenance space $O(k)$ [11] and with high probability an upper bound of $O(k * \log R)$ [11] for a k sized sample with replacement over a retention of R elements. It cannot be adapted to variable-size retentions. Priority sampling is proposed to overcome the issue by assigning each element a priority between 0 and 1. It has maintenance space $O(k * \log R)$ [11] both in expectation and as an upper bound with high probability for a k sized sample with replacement over R elements. The bounded space priority sampling technique produces samples without replacement. Storing the set of candidates S_c with timestamps in order and priorities in heap order, instead of in a simple array, improves the expected update time of the sample from $O(k * R)$ to $O(R + k * \log(k * \log R))$ [29] for a sample bounded in size k. Note that no lower bound on such a sample can be guaranteed, though at most k test and candidate items need to be stored so maintenance space seems to be $O(k)$.

4.5.3 Reservoir Sampling

Another technique to obtain a k sized sample is based on reservoir sampling. Reservoir sampling is a fixed size uniform sampling technique without replacement and does not require advance knowledge of R [2, 16, 30, 31, 62]. It uses a reservoir which has a capacity to store an unbiased sample of k elements. The initial $i \in [1...k]$ elements are immediately stored in the reservoir. As element $(i + 1)$ is processed, it has a probability of $\frac{k}{i+1}$ of being included in the reservoir and a probability of $\frac{1-k}{i+1}$ of being excluded. If it is included, it replaces a randomly selected old element in the reservoir. It can be proved by induction that this technique maintains uniformity [62]. The technique can be implemented very efficiently [1, 62]. To speed up the sampling instead of computing for each element the probability of inclusion in the sample, the random number of elements to skip is computed after an inclusion [16, 62]. The probability mass function for reservoir sampling with $k \geq 1$ is [16]:

$$\mathcal{P}(S; D) = \begin{cases} 1/\binom{|D|}{k} & \text{if |S| = k} \\ 0 & \text{otherwise} \end{cases}$$

While the fixed sample size is a major advantage of reservoir sampling, e.g. over the unbounded Bernoulli sampling and bounded space priority sampling, the technique does not fit into the incremental approach under consideration. It its current form, the reservoir sampling operates over an entire stream with no notion of retention. Imagine the reservoir of size k after the first k elements have arrived. For element $i + 1$ the probability of insertion is $\frac{k}{i+1}$, for element $i + 2$ the probability of insertion is $\frac{k}{i+2}$, etc. For fixed k and monotonically increasing i, the probability of a new element being inserted monotonically decreases over time. Intuitively later elements might not be reflected well in the sample. Instead the reservoir reflects the entire history since it was first maintained. A solution is non-obvious. One cannot create a reservoir sample over single element partitions, as either $k = 0$ or $k = 1$ causes none or all elements being stored respectively. Removing expired elements from the sample and replacing them with new elements, introduces a periodicity unacceptable for many applications and it breaks uniformity [11]. An alternative is to use a panes-like variant where a reservoir sample is created for each partition. The reservoirs of non-expired panes covering the retention can be combined into a single reservoir. A proper merge procedure is required [16], which involves more effort than merging Bernoulli samples.

Fortunately a solution to this problem is a technique called biased reservoir sampling [1]. A biased reservoir sample with size k at time of arrival of the t-th element has probability $p(r, t) = C * f(r, t)$ with $f(r, t) = e^{-\lambda*(t-r)}$ of the r-th element belonging to the sample ($r \leq t$). For simplicity, assume without loss of generality that $C = 1$, hence it can be omitted and $p(r, t) = f(r, t)$. This essentially states that at some r and increasing t the probability $p(r, t)$ decreases monotonically, i.e. the probability of an old element belonging to the sample decreases as new elements arrive. At some fixed t and increasing r the probability $p(r, t)$ increases monotonically, i.e. more recent elements have a higher probability of belonging to the sample. The memory-less exponential bias function $f(r, t)$ has a bias rate $\lambda \in [0.0, 1.0]$ and a probability of retaining a current element in the sample independent of the past or time of arrival ($p(r, t) = f(r, t) = 1.0$ as $r = t$). The bias function handles a relaxed retention requirement, i.e. it is not guaranteed that only the most recent elements are in the sample or that all old elements are replaced. The maximum reservoir size is a constant [1]:

$$\frac{1}{1 - e^{-\lambda}} \text{ which for sufficiently small } \lambda \text{ is about } \frac{1}{\lambda}$$

Simple formula rearrangement allows one to compute λ based on the sample size, although conversely a sensible λ can dictate the sample size. The technique of biased reservoir sampling works as follows:

1. Initialize an empty reservoir with size $\left\lceil \frac{1}{1 - e^{-\lambda}} \right\rceil$.
2. The fraction of the reservoir filled at the time of (just before) the arrival of the t-th element is $F(t) \in [0.0, 1.0]$.
3. The probability of $(t + 1)$-th element being included at time $(t + 1)$ is $p(r, t) = 1.0$ as $r = t + 1$, i.e. it is always included into the reservoir:
   (a) With probability F(t) the $(t+1)$-th element randomly replaces an old element in the reservoir.
   (b) Otherwise the $(t + 1)$-th element is inserted into the reservoir.
4. Repeat from second step for the next element.

Biased reservoir sampling is shown in figure 4.5 and illustrated in example 4.5.2.

Example 4.5.2. The choice of λ will affect the sample size and vice versa. Suppose there is a stream of elements as in figure 4.5 and one is interested in the average over the last 8 elements, sliding every 4 elements. The average is approximated by a sample of size $k = 4$ which is satisfied by $\lambda = 0.5$. For each element $t + 1$ in the first batch of data, the fraction of the reservoir filled at the time of (just before) the arrival of element t ($F(t)$) is determined. Initially the reservoir is empty causing the success probability $F(t)$ is 0 (failure), hence insertion will occur. The sample after processing the first element of the first batch of data is { 8 }. $F(t)$ causes failure for the second element too, the sample becomes { 8, 10 }. The third element 6 has a success probability $F(t) = \frac{1}{4}$. Upon success it randomly replaces an element in the

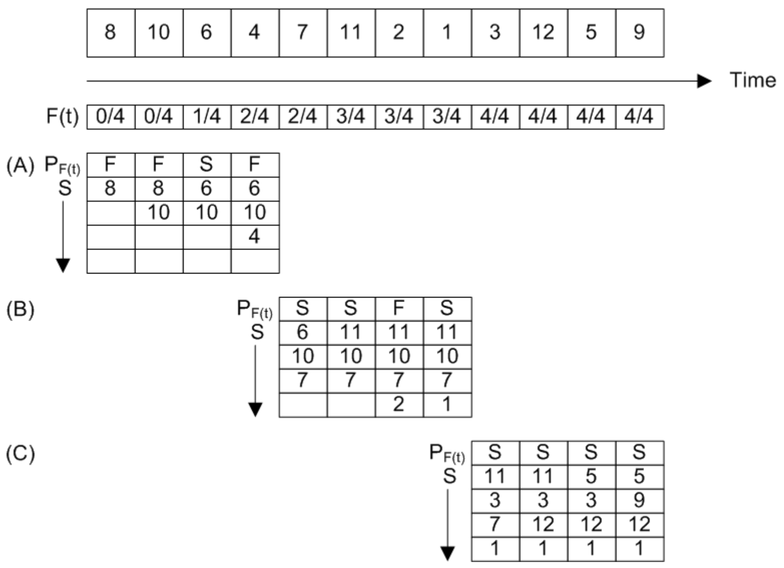


Figure 4.5: A biased reservoir sample of $k = 4$ with $\lambda = 0.5$. Every element has a success probability $P_{F(t)} = F(t)$ of randomly replacing an element. Otherwise it is inserted. The fraction at time t is denoted by $F(t)$ and shown per element. For $P_{F(t)}$ success S and failure F are shown per element. For each element the sample is shown vertically. At the first element 8 of the first batch of data is processed (a). It has $F(t) = 0$ and $P_{F(t)}$ a failure causing an insertion. The second element has $F(t) = 0$ and $P_{F(t)}$ a failure. The third element 6 replaces the element 8 in the sample. The fourth element is inserted. After the first batch the sample is $6, 10, 4$. The process continues for the other batches, (b) and (c). When $F(t) = k$ only replacements are possible. ○

sample, say 8. The fourth element of the first batch has success probability $F(t) = \frac{2}{4} = \frac{1}{2}$. If it has a failure, the element is inserted causing the sample to be of size 3. This process continues for the next batches of data. The averages yielded are $20/3 = 6\frac{2}{3}$, $29/4 = 7.25$ and $27/4 = 6.75$ for the first, second and third batch respectively. By the way, at the first element of the third batch $F(t) = k = 4$, thus only replacements will happen and no insertions as the reservoir is full. ◇

Note that the reservoir sampling need not store timestamps. The expiration is controlled probablistically by replacements and depends on the choice of λ. This contrasts with panes and expiration lists. An array can be used to keep the sample. The maintenance space of biased reservoir sampling is $O\left(\left\lceil \frac{1}{1-e^{-\lambda}} \right\rceil\right)$. The update time of the sample is $O(1)$ to randomly decide whether an insert or replace is performed. The notion of retention is relaxed as the most recent elements are in the sample with certain probability. It is not guaranteed only the most recent elements are in the sample and all expired elements (with respect to the retention) are replaced.

4.5.4 Final Note on Sampling

In summary there are quite some ways to perform sampling. In general sampling is generally applicable that can be used to approximate various aggregates. Its probablistic nature, trading accuracy for reduced memory space usage, lowers its robustness. For variance no ϵ-approximate sampling technique is found, however for quantiles a sample size of $O(\frac{1}{\epsilon^2} * \log \frac{1}{\epsilon * \delta})$ [5] should suffice. Biased reservoir sampling provides fixed sample sizes, while bounded space priority sampling without replacement provides only an upper bound on the sample size. Both can be employed for fixed-size and variable-size retentions. This does not hold for sequence-based sampling. Biased reservoir sampling is very efficient. It has a constant update time per new element and constant maintenance space. It has a relaxed notion of retention though. Its efficiency makes biased reservoir sampling an interesting candidate for the empirical part of the research (see chapters 5, 6 and 7). Suppose the sample size is bounded to $O(\epsilon^{-2} * \log(\epsilon * \delta)^{-1})$ for both variance and quantiles. Then variance under biased reservoir sampling takes:

- Maintenance space $O(\epsilon^{-2} * \log(\epsilon * \delta)^{-1} + 1) \sim O(\epsilon^{-2} * \log(\epsilon * \delta)^{-1})$, to keep the sample in an array and keep the internally maintained variance.

- Update time $O(1+1+1) \sim O(1)$, to include a new element in the sample and to undo an overwritten element from and apply a new element to the internally maintained variance.
- Query time $O(1)$, to lookup the internally maintained variance.

For quantiles under biased reservoir sampling it takes:

- Maintenance space $O(2 * \epsilon^{-2} * \log(\epsilon * \delta)^{-1}) \sim O(\epsilon^{-2} * \log(\epsilon * \delta)^{-1})$, to keep the sample in an array and keep a (balanced) binary search tree of the sample.
- Update time $O(1 + 2 * \log(\epsilon^{-2} * \log(\epsilon * \delta)^{-1})) \sim O(\log(\epsilon^{-2} * \log(\epsilon * \delta)^{-1}))$, to include a new element in the sample plus to undo an overwritten and insert a new element in the tree.
- Query time $O(\epsilon^{-2} * \log(\epsilon * \delta)^{-1})$, to sequentially scan the tree for the quantiles.

4.6 Histograms

Histograms have their uses in many places, among others in databases for query selectivity and in mathematics for data distribution approximation [44]. A histogram over some set of data is the partitioning of the data into one or more disjoint subsets referred to buckets or bins [34, 44]. Generally the buckets maintain certain statistics of interest, like the number of elements from the data it contains. The exact histogram has a very simple partitioning scheme: every unique value has a bucket. Computing aggregates is relatively easy, however exact histograms are unbounded aggregates themselves as summarized in table 3.1 in chapter 3. It is not hard to see that exact histograms require space proportional to the number of unique elements in the data. Another partitioning scheme has buckets representing fixed intervals, yielding equi-width histograms. Alternatively, equi-depth or equi-height histograms have a partitioning so that each bucket contains (approximately) the same number of elements. In contrast, a V-optimal histogram has a partitioning that tries to minimize the frequency variance of the different values in the buckets [2]. The partitioning affects the histogram being constructed and the approximation or error with respect to some error measure. In the context of a histogram being used as a synopsis structure, the aim is not to define the best (approximated) histogram, but rather histograms that approximate some aggregate well and support the notion of retention.

4.6.1 Exponential Histogram

An exponential histogram [27] basically stores each new element in a bucket along with a timestamp. To free up memory space buckets are merged when possible and expired buckets are removed. At any time all buckets but the last (oldest) have non-expired elements. The last bucket, as a consequence of merging and keeping the most recent timestamp of the buckets being merged, potentially contains expired elements. It must however contain at least a single non-expired element, otherwise the bucket would have been expired. The uncertainty in the actual number of non-expired elements in the last bucket causes an error, which should be bounded by ϵ. Note that the ϵ-approximate count is defined similar to ϵ-approximate variance earlier this chapter.

The exponential histogram is described analogous to literature [27]. Suppose elements are represented as a 1 or 0 respectively (e.g. by some criteria) and one is interested in maintaining the count of 1's over the last R elements with relative error bound ϵ. This effectively creates a binary stream. Let there be m buckets with C_i being the count of the i-th bucket. The first $m-1$ buckets have an exact count of 1's, while the last (oldest) bucket m has at least a count of 1. Consequently a lower bound on the total count of the buckets is $1 + \sum_{i=1}^{m-1} C_i$. The number of non-expired 1's in the last bucket is anywhere between 1 and C_m. Estimating it at $\frac{C_m}{2}$ limits the absolute error of the count of the last bucket to $\frac{C_m}{2}$. Clearly one wants the contribution of the last bucket to be small enough to maintain the error bound. For a relative error $\epsilon > 0$, let $k = \lceil \frac{1}{\epsilon} \rceil$ so that $\frac{1}{k} \le \frac{1}{\epsilon}$. The relative estimation error is at most:

$$\frac{\frac{C_m}{2}}{1 + \sum_{i=1}^{m-1} C_i}$$

To guarantee the error of the total count $\frac{1}{k} \le \frac{1}{\epsilon}$, the following invariant must hold:

$$\frac{\frac{C_m}{2}}{1 + \sum_{i=1}^{m-1} C_i} \le \frac{1}{k}$$

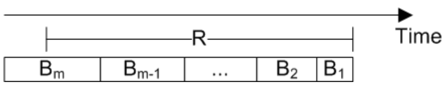


Figure 4.6: An exponential histogram over a binary stream. The oldest bucket B_m has some expired elements with respect to R, consequently causing the error in counting.

A second invariant must hold to keep the number of buckets low: at all times the bucket counts C_i are non-decreasing $C_1 \leq C_2 \leq ... \leq C_m$ and constrained to $\{1, 2, 4, ..., 2^{m'}\}$ for $m' \leq m$ and $m' \leq \log \frac{2*R}{k} + 1$. For every bucket but the last there are between $\frac{k}{2}$ and $\frac{k}{2} + 1$ buckets of the same count. It turns out that this second invariant implies the first invariant [27]. A schematic overview of m buckets counting over the last R elements is depicted in figure 4.6.

To estimate the total count of the exponential histogram, assume a counter T keeps the total count and L the count of the last bucket. Maintenance proceeds as follows [27]:

1. If the last bucket has expired:
   (a) Update T by decrementing it with the count of the last bucket.
   (b) Remove the last bucket.
   (c) Set L to the count of the current last bucket.
2. For a new 1 (as 0's are ignored), increment T and create a new bucket with count 1 and the current timestamp.
3. Traverse the buckets in order of increasing count:
   (a) If the number of buckets with the same count $\geq \frac{k}{2} + 1$, the two buckets with the oldest timestamp are merged into a single bucket with a double count and the most recent timestamp of the two.
   (b) If the last bucket was merged, update L with the new count of the last bucket.
4. The estimated total count is $T - \frac{1}{2} * L$.

During traversal a bucket resulting from a merge can in turn cause merges, hence merges can cascade and require a full pass over all buckets. The time for merging can be amortized by running it periodically, violating the second invariant temporarily and restoring it periodically. The estimate is the evaluation of a simple formula due to the maintenance of the counters T and L and thus is performed in constant time. By using real (clock) timestamps the exponential histograms work both for count and time-based windows. With little effort it can be made to work both for fixed-size and variable-size retentions. Exponential histogram maintenance is shown in figure 4.7 and illustrated in example 4.6.1.

Example 4.6.1. The relative error $\epsilon = 0.5$, so $k = 2$ causing mergers when $\frac{2}{2} + 2 = 3$ buckets have the same count. Suppose there is a stream of elements as in figure 4.7 and one is interested in the ϵ-approximate count of 1's over the last 8 elements, sliding every 4 elements. The first element of the first batch of 4 elements (a) causes a bucket to be created with count 1 and timestamp t_0. The total counter T and last bucket counter are set to 1. The second element creates another such bucket, with timestamp t_1, and increments only T. The third bucket is a 0 and can be ignored. The fourth bucket initially creates a bucket with count 1 and timestamp t_3. The counter T is incremented. As there are 3 buckets with the same count, the oldest two are merged. This causes bucket t_0 and t_1 to be merged into a bucket with count 2 and timestamp t_1. As the last bucket is the result of a merger, the counter L is set to that count (2). The second batch of 4 elements (b) starts with a 0 and is not inserted. Next a 1 causes a new bucket. Another 1 causes a merger. The last element causes a new bucket. The histogram now has a total count of 6, a last bucket count of 2. It consists of two buckets of count 2 and two buckets of count 1. The third batch of 4 elements (c) starts with a 0 that is not inserted. However the current last bucket is expired and removed, having the counter L set to the new last bucket count. Next an element of 1 causes a merger. Thereafter another one causes a new bucket. Finally a 1 causes a merger of two buckets of count 1 into a bucket of count 2. This is followed by two buckets of count 2 into one bucket of count 4 (cascading merge). The approximated count is $7 - 0.5 * 4 = 5$ which has a relative error of $\frac{7-5}{7} = \frac{2}{7} < \epsilon$. $\diamond$

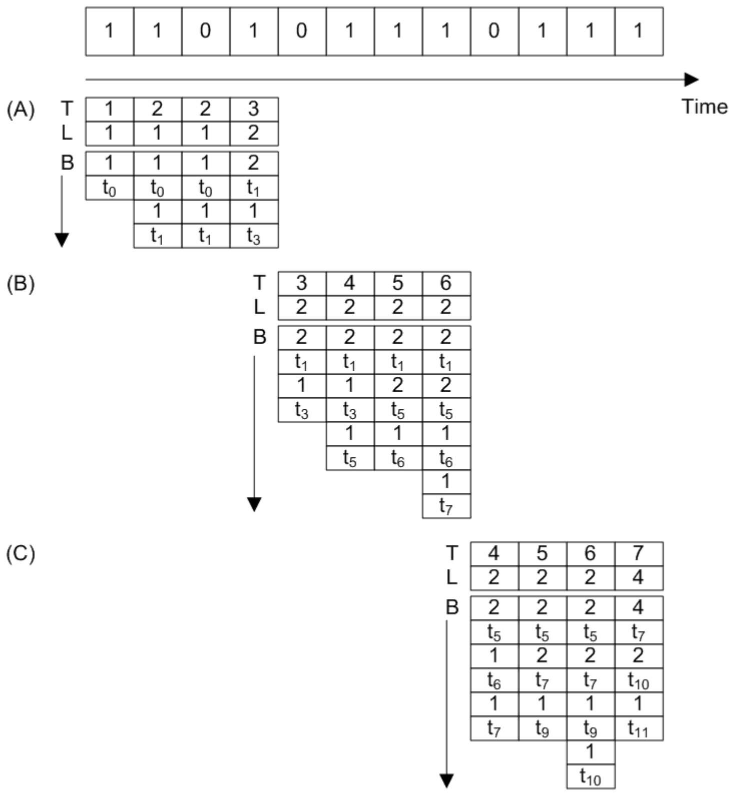


Figure 4.7: An exponential histogram over a binary stream with $\epsilon = 0.5$ and $k = 2$. Per element T is the total count, L the last bucket's count and from B downwards the buckets with count and timestamp. The first element causes a bucket with count 1 and timestamp t_0 and consequently $T = 1$ and $L = 1$. The second element creates another bucket and increases the total. The third element is not inserted as it is a 0. The fourth element causes the third bucket with a count of 1, hence the oldest two buckets with a count of 1 are merged into a bucket with count 2 and their most recent timestamp. And so on. ◦

Exponential histograms can be extended to support sums of positive integers in the range $[0...N]$ and by keeping a histogram for the count along with it, the average can be computed with small relative error (it is the quotient of small relative error estimates) [27]. Other uses can be imagined. For example keeping an exponential histogram for an approximate count for each bucket of an equi-width histogram. In [57] it is used to maintain the approximate counts of buckets in dynamic equi-depth histograms where the boundaries of the equi-depth buckets evolve with the data distribution. However no formal error bound is guaranteed for those histograms. In general exponential histograms are capable of supporting any function f that for all multisets X, Y is [9, 27]:

- Positive, $f(X) \geq 0$.
- Polynomially bounded, $f(X) \leq poly(|X|)$.
- Composable, $f(X \cup Y) \geq f(X) + f(Y)$.
- Weakly additive, $f(X \cup Y) \leq C * (f(X) + f(Y))$ where the constant $C \geq 1$.
- For efficiency $f(x)$ must be computable (approximated) by a small synopsis which is additive.

In summary, exponential histograms as discussed above can be used to obtain the ϵ-approximate count of 1's over the last R elements (in a binary stream). As argued for in chapter 3 when sliding 1 element at a time, the only way to reduce memory space usage is by approximation. This is where exponential histograms fit in. To compute the ϵ-approximate count under exponential histograms it takes:

- Maintenance space $O(\frac{1}{\epsilon} * \log^2 R)$ [27].
- Update time $O(\log R)$ or amortized $O(1)$ [27].
- Query time $O(1)$ [27].

Exponential histograms are not as broadly applicable as expiration lists, panes and sampling because the functions f are constrained. It is not able to handle variance [9] or quantiles. However, the histograms are expanded upon in the next subsection to obtain an aggregate tailored synopsis structure for approximating variance. Compared to sampling, exponential histograms are robust as they guarantee an upper bound on the relative error.

4.6.2 Variance Histogram

As stated above, an aggregate like variance cannot be estimated by exponential histograms [9]. However, a slight revision / extension of the exponential histograms provides variance histograms to compute the ϵ-approximate variance last R elements [9]. Similar to exponential histograms, buckets are maintained, but over real valued elements. Suppose there are m buckets with B_m denoting the last (oldest) bucket. A bucket B_i stores for its elements the oldest timestamp t_i, count n_i, average μ_i and the variance as the sum of squared differences V_i. A suffix bucket B_i* represents all elements in the stream that arrived after the elements of B_i . The suffix bucket is defined as $B_i* = \bigcup_{i=j}^{i-1} B_j$. The combination of buckets B_i and B_j into $B_{i,j}$ happens by the incremental formula [9, 12, 20]. The combined bucket has the most recent timestamp of the buckets being combined. The formulae are shown respectively for the count, average and variance (as the sum of squared differences):

$$n_{i,j} = n_i + n_j \tag{4.1}$$

$$\mu_{i,j} = \frac{\mu_i * n_i + \mu_j * n_j}{n_{i,j}} \tag{4.2}$$

$$V_{i,j} = V_i + V_j + \frac{n_i * n_j}{n_{i,j}} * (\mu_i - \mu_j)^2 \tag{4.3}$$

Rearranging the formula allows one to compute the decremental variants, e.g. $n_i = n_{i,j} - n_j$, that can be used to separate B_i from $B_{i,j}$ to obtain B_j. A schematic overview similar to the one for exponential histograms is shown in figure 4.8. The last bucket B_m has an expired and non-expired part. The non-expired part is denoted by B_{m^-}. Only the suffix bucket B_m* for bucket B_m is shown.

The following procedure describes the maintenance of the variance histogram [9]:

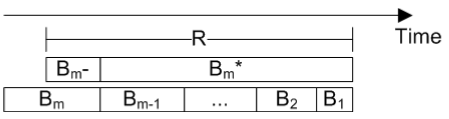


Figure 4.8: A variance histogram, the oldest bucket B_m has some expired elements with respect to R, consequently causing the error in computing the variance. The suffix bucket B_m* is the combination of all buckets after B_m, hence representing the elements that arrived thereafter. The bucket B_{m^-} denotes the non-expired part of bucket B_m. ∘

1. If the i-th element equals the average of the most recent bucket B_1, then increment the count n_i of the bucket.
2. Else B_i becomes B_{i+1} and a new bucket B_1 is created with timestamp t_1 equal to the current time, count $n_1 = 1$, average μ_1 equal to the value of the i-th element and variance $V_1 = 0$.
3. Update B_m* for the new element ($n_1 = 1$, μ_1 equal to the value of the i-th element and $V_1 = 0$) using the incremental formula.
4. Starting from bucket B_m until no expired bucket is found, remove B_m so that B_{m-1} is the last bucket. Maintain $B_{m-1}* = B_m * \backslash B_{m-1}$ by applying the decremental formula with B_m* and B_{m-1} to obtain $B_{m-1}*$.
5. Let $V_{i,i-1}$ be the variance after combining B_i and B_{i-1} and $k = \frac{9}{\epsilon^2}$.
6. While there exists an index $i > 2$ such that $k * V_{i,i-1} \le V_{i-1}*$, use the incremental formula to combine buckets B_i and B_{i-1} for the smallest such i.

The third step combines B_m* with a the new bucket B_1, regardless of whether the first or second step was performed previously. Suppose the buckets are stored in a sequence. The first bucket is the most recent one and the last bucket is the oldest one (with the oldest timestamp). The fourth step can exploit the ordering by starting at the back of the sequence going to the front until the first non-expired bucket is found. The traversal in the sixth step starts from the third bucket, $i = 3$, having $B_3* = B_1 \cup B_2$. As the traversal proceeds $B_i* = B_{i-1} * \cup B_{i-1}$. Note that after the traversal is complete one ends up with B_m* for free. Note that the suffix buckets other than B_m* are only used and computed during the sixth step. The time for merging can be amortized by running the sixth step periodically.

It can be shown that the relative error of variance is within ϵ by maintaining the invariant: for every bucket B_i it holds that $\frac{9}{\epsilon^2} * V_i \le V_i*$ where V is the variance of the last R elements [9]. This invariant is maintained by the procedure above, because buckets have non-zero variance only when they are combined and the condition for combining them preserves the invariant for the combined bucket. The invariant will hold for B_i over time as the suffix bucket B_i* is non-decreasing over time. A second invariant keeps the number of buckets small: for each $i > 1$, for every bucket B_i it holds that $\frac{9}{\epsilon^2} * V_{i,i-1} > V_{i-1}*$ [9]. The maintenance procedure is illustrated in example 4.6.2.

Example 4.6.2. Suppose a stream of elements $8, 10, 6, 4, 7, 11, 2, 1, 3, 12, 5$ and 9 with timestamps t_0 to t_{11} respectively. One is interested in the ϵ-approximate variance over the last 8 elements, sliding every 4 elements. Hence there are three batches $\{8, 10, 6, 4\}$, $\{7, 11, 2, 1\}$ and $\{3, 12, 5, 9\}$. Assume $\epsilon = 0.5$ and $k = \frac{9}{\epsilon^2} = 36$. The table below shows the variance histogram after the first three elements are inserted.

$\mathbf{B}_i$					$\mathbf{B}_{i-1}*$			$\mathbf{B}_{i,i-i}$	
i	t_i	n_i	μ_i	V_i	$n_{i-1}*$	$\mu_{i-1}*$	$V_{i-1}*$	$V_{i,i-1}$	$k * V_{i,i-1}$
1	2	1	6.00	0.00	0	0.00	0.00	0.00	0.00
2	1	1	10.00	0.00	1	6.00	0.00	8.00	288.00
3	0	1	8.00	0.00	2	8.00	0.00	2.00	72.00

The first element 8 of the first batch becomes a bucket with timestamp $t = 0$, count $n = 1$, average $\mu = 8$ and variance $V = 0$. The second element 10 of the first batch becomes a bucket with $t = 1$, $n = 1$, $\mu = 10$ and $V = 0$. Up until this point $B_{i-1}*$ and $B_{i,i-1}$ have no use. The third element 6 of the first batch becomes a bucket with $t = 2$, $n = 1$, $\mu = 6$ and $V = 0$. Now a potential merger can be searched for. The third bucket (created for the first element inserted) currently is the first and only one to have a suffix, $B_{i-1}*$ for $i = 3$ is $B_2* = B_1$. Furthermore $B_{i,i-1}$ is the combination of buckets B_3 and B_2,

$B_{3,2} = B_3 \cup B_2$. The condition $k * V_{i,i-1}$ is $36 * V_{3,2} \leq V_2* = V_1$ does not hold and thus merging is not possible. The table below shows the variance histogram state after the next 5 elements are inserted; the first and second batch are processed.

$\mathbf{B}_i$					$\mathbf{B}_{i-1}*$			$\mathbf{B}_{i,i-i}$	
i	t_i	n_i	μ_i	V_i	$n_{i-1}*$	$\mu_{i-1}*$	$V_{i-1}*$	$V_{i,i-1}$	$k * V_{i,i-1}$
1	7	1	1.00	0.00	0	0.00	0.00	0.00	0.00
2	6	1	2.00	0.00	1	1.00	0.00	0.50	18.00
3	5	1	11.00	0.00	2	1.50	0.00	40.50	1458.00
4	4	1	7.00	0.00	3	4.67	0.50	8.00	288.00
5	3	1	4.00	0.00	4	5.25	60.67	4.50	162.00
6	2	1	6.00	0.00	5	5.00	64.75	2.00	72.00
7	1	1	10.00	0.00	6	5.17	66.00	8.00	288.00
8	0	1	8.00	0.00	7	5.86	66.83	2.00	72.00

As the element 3 is inserted, the bucket for element 8 expires as it was inserted before the last 8 elements (including element 3). Similarly, as element 12 is inserted, the bucket for element 10 expires. The expired buckets are simply removed. The resulting variance histogram state is shown in the table below.

$\mathbf{B}_i$					$\mathbf{B}_{i-1}*$			$\mathbf{B}_{i,i-i}$	
i	t_i	n_i	μ_i	V_i	$n_{i-1}*$	$\mu_{i-1}*$	$V_{i-1}*$	$V_{i,i-1}$	$k * V_{i,i-1}$
1	9	1	12.00	0.00	0	0.00	0.00	0.00	0.00
2	8	1	3.00	0.00	1	12.00	0.00	40.50	1458.00
3	7	1	1.00	0.00	2	7.50	0.00	2.00	72.00
4	6	1	2.00	0.00	3	5.33	40.50	0.50	18.00
5	5	1	11.00	0.00	4	4.50	68.67	40.50	1458.00
6	4	1	7.00	0.00	5	5.80	77.00	8.00	288.00
7	3	1	4.00	0.00	6	6.00	110.80	4.50	162.00
8	2	1	6.00	0.00	7	5.71	112.00	2.00	72.00

From the table above it is clear that two merges are possible:

1. $B_4 \cup B_3$ with timestamps 6 and 7 into a bucket with timestamp 7, because $36 * V_{4,3} = 18 \leq V_3* = 40.5$ as obvious from the table at $i = 4$.

2. $B_8 \cup B_7$ with timestamps 2 and 3 into a bucket with timestamp 3, because $36 * V_{8,7} = 72 \leq V_7* = 112$ as obvious from the table at $i = 8$.

After merging, the element 5 is inserted. Without merges, the bucket for element 6 would have expired. However as it is merged away, no expiration happens. Next element 9 is inserted and the oldest bucket is expired. The final variance histogram state after all 12 elements have been inserted is shown in the table below.

$\mathbf{B}_i$					$\mathbf{B}_{i-1}*$			$\mathbf{B}_{i,i-i}$	
i	t_i	n_i	μ_i	V_i	$n_{i-1}*$	$\mu_{i-1}*$	$V_{i-1}*$	$V_{i,i-1}$	$k * V_{i,i-1}$
1	11	1	9.00	0.00	0	0.00	0.00	0.00	0.00
2	10	1	5.00	0.00	1	9.00	0.00	8.00	288.00
3	9	1	12.00	0.00	2	7.00	0.00	24.50	882.00
4	8	1	3.00	0.00	3	8.67	8.00	40.50	1458.00
5	6	2	1.50	0.50	4	7.25	24.67	2.00	72.00
6	5	1	11.00	0.00	6	5.33	48.75	60.67	2184.00
7	4	1	7.00	0.00	7	6.14	93.33	8.00	288.00

This concludes the example on variance histogram maintenance. ◇

With the exception of the last bucket B_m, all other buckets cannot contain expired elements. Let B_{m^-} be the non-expired part of the bucket B_m. This is shown schematically in figure 4.8. The estimated count, average and variance can be computed from $B_m * \cup B_{m^-}$. However the count, average and variance need to be estimated for B_{m^-} first. The following procedure describes the variance estimation from the histogram [9]:

1. For the non-expired part of B_{m^-} estimate the count (n_{m^-}), average (μ_{m^-}) and variance (V_{m^-}):
   (a) $n_{m^-} = R - n_m*$
   (b) $\mu_{m^-} = \mu_m$
   (c) $V_{m^-} = \frac{1}{2} * V_m$
2. $\widehat{V} = V_{m^-} + V_m * + \dfrac{n_{m^-} \cdot n_m*}{n_{m^-} + n_m*} \cdot (\mu_{m^-} - \mu_m*)^2$
3. The estimate for actual variance $\widehat{\sigma^2} = \dfrac{\widehat{V}}{n_{m^-} + n_m*}$ and for standard deviation $\widehat{\sigma} = \sqrt{\widehat{\sigma^2}}$.

From the estimation procedure it is clear why suffix bucket B_m* should be stored. By suffix bucket B_m* the estimates for B_{m^-} can be performed in constant time. Consequently the combination of the buckets to obtain the final variance and/or standard deviation estimates are performed in constant time too. Note that the count estimate actually is exact for fixed-size retentions [9]. The estimation procedure is illustrated in example 4.6.3.

Example 4.6.3. In example 4.6.2 the maintenance is shown. An estimate of the variance over the last 8 elements is obtained from from the last table of example 4.6.2, showing the state of the variance histogram after the 12 elements are inserted. To this end B_m* and B_{m^-} are required. Note that B_m* is the combination of all buckets $i = 1, 2, ..., m-2, m-1$. It can be obtained from the last table of example 4.6.2 where $m = 7$, by combining the values for $B_{i-1}*$ on the last row with the values of B_i in the one but last row. This yields B_m* as $B_7* = B_6 * \cup B_6$:

1. $n_m* = 7 + 1 = 8$
2. $\mu_m* = \dfrac{7 * 6.14 + 1 * 11}{7 + 1} \approx 6.75.$
3. $V_m* = 93.33 + 0 + \dfrac{7 * 1}{7 + 1} * (6.14 - 11)^2 \approx 114.$

Clearly, after a full pass through the buckets in step 6 of the maintenance algorithm provides B_m* for free. The values for B_{m^-} with $m = 7$ can be estimated from the last row of the last table of example 4.6.2:

1. $n_{m^-} = 8 - 8 = 0$
2. $\mu_{m^-} = 7.$
3. $V_{m^-} = \dfrac{0}{2} = 0.$
4. $\widehat{V} = 0 + 114 + \dfrac{0 * 8}{0 + 8} * (7 - 6.75)^2 \approx 114.$ The actual (population) variance over the last 8 elements is $\widehat{\sigma^2} = \dfrac{114}{8} = 14.25.$

The actual (population) variance is 15.19, hence the estimate has a relative error far less than $\epsilon = 0.5$. Note that values are shown rounded to at most two decimals, however computations use the more precise values. $\diamond$

Note that the procedures work well for fixed-size retentions. Supporting variable-size retentions is non-obvious. The number of elements R in the retention can vary per retention. This requires a means to count the elements in the retention, which is computationally expensive to do exact (see the discussion of such problems in chapter 3). Exponential histograms could be used to maintain an ϵ-approximate count of the retention. Or similarly to exponential histograms, the count of the last bucket is estimated to be half the count maintained by the bucket. If the last bucket's contribution is sufficiently small, the estimate might still be effective. Neither proposed solution has currently been proved to maintain the ϵ-approximate variance!

In summary, a variance histogram reuses the ideas behind exponential histogram to obtain the ϵ-approximate variance over a retention of R elements. Its adaptation to support variable-size retentions is non-obvious. As argued for in chapter 3 when sliding 1 element at a time, the only way to reduce memory space usage is by approximation. This is where variance histograms fit in. To compute ϵ-approximate variance under a variance histogram it takes:

- Maintenance space $O(\frac{1}{\epsilon^2}*\log R)$ [9], proportional to the number of buckets in the variance histogram.
- Update time $O(\frac{1}{\epsilon^2}*\log R)$ or amortized $O(1)$ [9], as the merge step takes time proportional to the number of buckets and is amortized by running it periodically.
- Query time $O(1)$ [9], to estimate the variance.

An optimized version of variance histograms is described in literature [65]. Note that the idea behind variance histograms need not be limited to variance [9]. In its current form it has limited applicability as it is tailored to the aggregate variance. The synopsis structure is robust as it provides ϵ-approximate variance, which has clear benefits over sampling.

4.6.3 Final Note on Histograms

Maintaining exact histograms can be very space expensive if the number of unique elements is large. An exponential histogram provides a means for ϵ-approximate counting over the last R elements in a binary stream and can be extended to support other aggregates over other kinds of stream. However, it fails to support aggregates like variance. The variance histogram extends upon the exponential histogram by providing ϵ-approximate variance over the last R elements. It is tailored to variance, yet its adaptation to variable-size retentions is non-obvious. An advantage of exponential and variance histograms are their memory space savings when sliding 1 element at a time. In contrast, expiration lists and panes provide no such benefit. An advantage of the histograms over sampling is their deterministic achievement of the relative error bound.

Limited literature was found on the use of histograms to compute aggregates in sliding windows. Exponential histograms are used for more complex histograms [57], but no error bounds are provided. Error bounds on histograms with respect to some error measure have been proposed [40], however the error measure is about the histogram and not the aggregates computed from it. As a consequence of implicit deletes in the incremental approach, proposed histograms with explicit deletes and/or backing samples [34] are not useful and they do not seem to bound the error of the aggregates.

4.7 Quantile Summary

The difficulty with quantiles is that they belong to the unbounded (see definition 3.4.3) and non-differential (see definition 3.4.4 aggregates as discussed in chapter 3. Deterministic computation of exact quantiles over R elements in P passes requires $O(R^{\frac{1}{P}})$ space [54]. Trading accuracy for space seems the only way to reduce memory space usage. Note how ϵ-approximates are defined differently for quantiles and variance, as defined at the beginning of this chapter.

Synopsis structures to compute ϵ-approximate ϕ-quantiles over a stream are known from literature [26, 39]. However, the notion of retention and implicit deletes is not supported. In literature [5, 49] a few synopsis structures are described that are capable thereof. In this section the synopsis structure to compute ϵ-approximate ϕ-quantiles over the retention with the lowest maintenance space is described [5]. In contrast to another structure [49], it does not require advance knowledge of the universe from which the elements are drawn [5]. Interestingly the selected synopsis structure uses intervals that are aggregated, related to what is described in section 1.2.2 of the introductory chapter 1 on alternatives to the incremental approach. As it depends on a synopsis structure used on streams, that structure is described first. Thereafter the synopsis structure for the incremental approach is discussed.

4.7.1 Greenwald-Khanna (GK)

To compute ϵ-approximate ϕ-quantiles over streams, a synopsis structure referred to as Greenwald-Khanna (GK) is discussed [39]. At a conceptual level GK puts each element from the stream into its summary and periodically merges parts of the summary to free up space. The summary $S(n)$ after processing n elements is represented by tuples $t_i = \langle v_i, g_i, \Delta_i \rangle$. The value v_i corresponds to the value of an element. The g_i is $r_{min}(v_i) - r_{min}(v_{i-1})$, the difference between the minimum rank of v_i and v_{i-1}. The Δ_i is $r_{max}(v_i) - r_{min}(v_i)$, the difference between the maximum and minimum rank of v_i. The summary maintains the tuples in order of v_i. The minimum rank $r_{min}(v_i) = \sum_{j \leq i} g_j$ and the maximum rank $r_{max}(v_i) = \sum_{j \leq i} g_j + \Delta_i$. Moreover, $g_i + \Delta_i - 1$ is the upper bound on the total number of elements between v_{i-1} and v_i. The number of elements processed in the summary is $n = \sum_i g_i$. The quantile summary $S(n)$ is maintained so that:

$$e = \frac{\max_i(g_i + \Delta_i)}{2} \leq \epsilon * n$$

This ensures that value v_i with rank $r - e \leq r_{min}(v_i)$ and $r_{max}(v_i) \leq r + e$ is an ϵ-approximate ϕ-quantile [39]. It leads to the following procedure to obtain an ϵ-approximate ϕ-quantile: find v_i from the summary $S(n)$ so that $r - r_{min}(v_i) \leq \epsilon * n$ and $r_{max}(v_i) - r \leq \epsilon * n$. The procedure is illustrated in example 4.7.2. First a procedure to process elements and insert them into the summary is required.

The insert procedure is simple. For an element with value v insert a tuple between the tuples t_i and t_{i-1} in the summary $S(n)$ to obtain $S(n+1)$, so that $v_{i-1} \leq v < v_i$. If v is the new minimum or maximum the tuple $\langle v, 1, 0\rangle$ is inserted, otherwise $\langle v, 1, \lfloor 2 * \epsilon * n \rfloor\rangle$ is inserted between tuples t_{i-1} and t_i. Note that as n increases, $\lfloor 2 * \epsilon * n \rfloor$ is non-decreasing and hence later observations will have higher values for Δ_i. The summary can be represented as a (balanced) binary search tree of tuples t_i ordered on v_i.

To free up space, tuples with small capacity are merged into tuples with similar or larger capacities. To this end the values of Δ are partitioned into bands, where larger values of Δ have a lower capacity and vice versa. It is claimed that "bands are defined such that whenever two Δs are ever in the same band, they never appear in different bands as n increases" [39]. Let α go from 1 to $\lceil \log 2 * \epsilon * n \rceil$ and $p = \lfloor 2 * \epsilon * n \rfloor$, then $band_\alpha$ is the set of Δ in:

$$p - 2^\alpha - (p \bmod 2^\alpha) < \Delta \leq p - 2^{\alpha-1} - (p \bmod 2^{\alpha-1})$$

The $band_\alpha$ is defined to be p. As a special case, the first $\frac{1}{2*\epsilon}$ elements end up in a band of their own. The band of Δ_i after n elements have been processed is $band(t_i, n)$. A tuple can be considered full when $g_i + \Delta_i = \lfloor 2 * \epsilon * n \rfloor$ by:

$$\frac{\max_i(g_i + \Delta_i)}{2} \leq \epsilon * n \text{ or alternatively } \max_i(g_i + \Delta_i) \leq 2 * \epsilon * n$$

Relating to the capacities, two tuples can be merged when either has a similar or larger band and merging does not cause the merged tuple to be full. The compress procedure is as follows:

- For i from $|S(n)| - 2$ to 0 (from the one but last until the first tuple):
  If $band(t_i, n) \leq band(t_{i+1}, n)$ and $g_i + g_{i+1} + \Delta_{i+1} < 2 * \epsilon * n$, then merge t_i into t_{i+1} as $t_{i+1} = \langle v_{i+1}, g_i + g_{i+1}, \Delta_i + 1\rangle$ and deleting t_i.

Note that $2 * \epsilon * n$ increases as more elements are processed. Hence tuples can become mergeable as more elements are in the summary. The compression procedure above is a simplification. It lacks an additional tree structure imposed over the tuples, with children arranged in non-increasing order of the band. Literature [39] contains details. As a final remark, the compression is performed periodically (once every $\frac{1}{2*\epsilon}$ elements processed) to amortize its processing time. The procedures for insertion and compression are illustrated in example 4.7.1.

Example 4.7.1. Suppose $\epsilon = 0.2$, $n = 9$ and the the following elements have been processed by GK: 2, 4, 100, 44, 76, 49, 10, 98 and 27. This leads to the following summary:

v_i	2	4	10	27	44	49	76	98	100
g_i	1	1	1	1	1	1	1	1	1
Δ_i	0	0	2	3	1	2	1	2	0

Let the next element be 38, which is neither a minimum or maximum given the previously processed elements. The tuple inserted is $t = \langle 38, 1, \lfloor 2 * 0.2 * 9 \rfloor\rangle = \langle 38, 1, 3\rangle$, incrementing n by 1 after insertion. This leads to the following summary:

v_i	2	4	10	27	38	44	49	76	98	100
g_i	1	1	1	1	1	1	1	1	1	1
Δ_i	0	0	2	3	3	1	2	1	2	0

When a next element is processed, compression is performed first because $n \bmod \frac{1}{2*\epsilon} = 0$. Traversing the summary in reverse, the first point of interest is in comparing the tuples for 49 and 76 (denoting t_i and t_{i+1} respectively). Both have a band of 2 and hence satisfy the condition: $band(t_i, n) \leq band(t_{i+1}, n)$. Moreover $1+1+1 = 3 < 2*0.2*10 = 4$, hence condition $g_i + g_{i+1} + \Delta_{i+1} < 2*\epsilon*n$ is satisfied too. This leads to the tuple for 49 being removed from the summary and the tuple for 76 being updated to $\langle 76, 2, 1\rangle$. The next point of interest is in comparing the tuples for 38 and 44. The former has a band of 1 and the latter a band of 2, hence satisfying the first condition for compression. Moreover $1+1+1 = 3 < 4$, hence the second condition is satisfied as well. This leads to the tuple for 38 being removed from the summary and the tuple for 44 being updated to $\langle 44, 2, 1\rangle$. Compression causes no further changes. The summary now is:

v_i	2	4	10	27	44	76	98	100
g_i	1	1	1	1	2	2	1	1
Δ_i	0	0	2	3	1	1	2	0

Let the next element be 35, causing tuple $\langle 35, 1, \lfloor 2*0.2*10 \rfloor\rangle = \langle 35, 1, 4\rangle$ to be inserted. This leads to the following summary:

v_i	2	4	10	27	35	44	76	98	100
g_i	1	1	1	1	1	2	2	1	1
Δ_i	0	0	2	3	4	1	1	2	0

This concludes the example for inserting elements into the summary and its compression. ◇

Example 4.7.2. Suppose one wants to compute the median with $\phi = 0.5$ under $\epsilon = 0.2$ after $n = 11$ elements. Assume the summary is equal to the last table in the previous example 4.7.1; elements inserted were $2, 4, 100, 44, 76, 49, 10, 98, 27, 38$ and 35. The rank of the median is $r = \lceil \phi*n \rceil = 6$. The quantile is v_i where $r - r_{min}(v_i) \leq \epsilon*n$ and $r_{max}(v_i) - r \leq \epsilon*n$. The values for $r_{min}(v_i)$ and $r_{max}(v_i)$ are shown below:

v_i	2	4	10	27	35	44	76	98	100
$r_{min}(v_i)$	1	2	3	4	5	7	9	10	11
$r_{max}(v_i)$	1	2	5	7	9	8	10	12	11

The condition is first satisfied for $v_i = 27$. Assuming a dataset with the elements sorted, the exact rank for $v_i = 27$ is 4. The exact median with rank 6 is element 38. Clearly $\frac{6-4}{11} \leq 0.2$, hence the relative error bound is satisfied. ◇

In summary, the synopsis structure GK can compute ϵ-approximate ϕ-quantiles over n elements (e.g. a stream), but does not fit in the incremental approach in its current form as it lacks a notion of retention and implicit deletes. To compute ϵ-approximate ϕ-quantiles under GK it takes:

- Maintenance space $O(\frac{1}{\epsilon} * \log \epsilon n)$ [39], it is proportional to the number of tuples in the summary $S(n)$ after processing n elements.
- Update time is not mentioned in literature, however it consists of inserting a new tuple in the summary taking time logarithmic in the size of a (balanced) binary search tree of $O(\frac{1}{\epsilon} * \log \epsilon n)$ tuples. Note that compression takes amortized $O(1)$ and hence does not affect the previous.
- Query time is not mentioned in literature, however it requires at most a full pass over the tuples in the summary, hence it takes time $O(\frac{1}{\epsilon} * \log \epsilon n)$.

Quantiles encompass among others the minimum ($\phi = 0$), median ($\phi = 0.5$) and maximum ($\phi = 1.0$). Even though GK is tailored to quantiles, it provides some well known aggregates "for free". The synopsis structure is not as broadly applicable as some of the other synopsis structures in this chapter. It is robust as the relative error is bounded. In the next section GK is used (as a blackbox) by a synopsis structure that does fit in the incremental approach.

4.7.2 Arasu-Manku (AM)

To compute ϵ-approximate ϕ-quantiles in the incremental approach, a synopsis structure referred to as Arasu-Manku (AM) [5] is discussed. At the lowest level, elements belong to consecutive disjoint intervals over which GK is run. At a higher level, GK is run over intervals spanning two intervals of the level

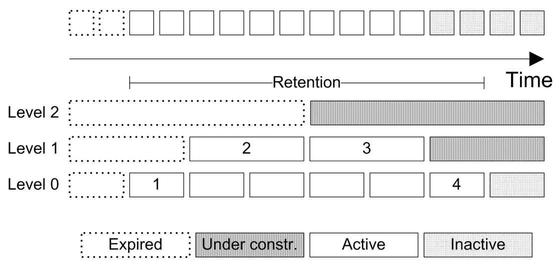


Figure 4.9: A stream with elements above the time axis. A retention determines elements and blocks that are expired (dotted lines), active (blank fill), under construction (densely dotted fill) and inactive (sparsely dotted fill). There are 3 levels. The blocks with numbers 1, 2, 3 and 4 are a disjoint union covering the retention. ∘

below. Each interval stores a number of quantiles, called a quantile sketch. A disjoint union of intervals covering the retention has its quantile sketches used to determine the ϵ-approximate ϕ-quantiles over the retention of R elements.

The synopsis structure is maintained as follows. The stream is partitioned into levels l from 0 to $L = \log \frac{4}{\epsilon}$. Each level is divided into consecutive disjoint intervals of the same size covering a number of elements, called blocks. A level l block has a size $R_l = \frac{\epsilon * R}{4} * 2^l$. So a block at level $l+1$ is the disjoint union of two blocks at level l. Blocks can be in any of the following states:

- Inactive if none of its element are processed (i.e. have arrived in the stream).
- Under construction if some of its elements are processed and none of its elements are expired.
- Active if all of its elements are processed and none of its elements are expired.
- Expired if at least one of its elements is expired.

Each level l has a relative error $\epsilon_l = \frac{\epsilon}{2*(2*L+2)} * 2^{L-l}$. Note that higher levels, have smaller error. Each block of a level l runs GK with relative error bound $\frac{\epsilon_l}{2}$ over its elements. When the block becomes active, the quantile summary of GK over the block is queried for all $\frac{\epsilon_l}{2}$-approximate quantiles with $\phi \in \{\epsilon_l, 2 * \epsilon_l, ..., 1\}$. The result is stored as the quantile sketch of a block. Any expired block is discarded. The synopsis obtained by AM over a fixed-size retention of R elements is denoted $F(R, \epsilon)$; an Arasu-Manku (AM) fixed sketch. The concepts are illustrated in figure 4.9.

To obtain the ϵ-approximate ϕ-quantiles, a disjoint union of active blocks is probed for their quantile sketches. Such a disjoint union can easily be obtained by traversing from the highest level to the lowest level, selecting each active block not already covered at a higher level (as shown in figure 4.9). Suppose the b blocks in the disjoint union have size $R_1, R_2, ..., R_b$ and error level $\epsilon_1, \epsilon_2, ..., \epsilon_b$. For each block i, the quantiles in the sketch of that block are associated a weight $\epsilon_i * R_i$. Of all sketches, the quantiles with their weights are put in a collection sorted on quantile value. An ϵ-approximate ϕ-quantile is obtained by the picking the quantile in that collection so that the sum of all weights preceding it is $< \lceil \phi * (R_1 + R_2 + ... + R_b) \rceil$ and the sum including the weight is $\geq \lceil \phi * (R_1 + R_2 + ... + R_b) \rceil$. It is claimed this is correct [5].

Note that the discussion above is relates to fixed-size retentions. An extension to variable-size retentions requires some changes [5] and provides the Arasu-Manku (AM) variable sketch. The idea is to maintain a collection of AM fixed sketches: $V(R, \epsilon) = \{F(2^k, \frac{\epsilon}{2}), F(2^{k-1}, \frac{\epsilon}{2}), ..., F(\frac{2}{\epsilon}, \frac{\epsilon}{2}\}$ where $2^{k-1} \leq R < 2^k$. At most $\lfloor \log(\epsilon * R) \rfloor$ of those AM fixed sketches are maintained. In a time-based context, the timestamp of the first element of a block should be stored by the block. Any block whose timestamp is expired, indicates an expired block.

Whenever a new element is inserted into $V(R, \epsilon)$, all AM fixed sketches in the collection have the new element inserted. Suppose $r+1$ is the number of elements in $F(2^k, \frac{\epsilon}{2})$ after insertion. If $2^k + 1 = r + 1$, then $F(2^{k+1}, \frac{\epsilon}{2})$ is created from $F(2^k, \frac{\epsilon}{2})$. For all but the new AM fixed sketch $F(2^{k+1}, \frac{\epsilon}{2})$, the oldest element is removed. Upon deletion of an element, the oldest element in $F(2^k, \frac{\epsilon}{2})$ is removed. Suppose $r - 1$ is the number of elements covered in $F(2^k, \frac{\epsilon}{2})$. If $2^{k-1} = r - 1$, then $F(2^k, \frac{\epsilon}{2})$ is redundant with respect to $F(2^{k-1}, \frac{\epsilon}{2})$. The former AM fixed sketch is discarded.

To retrieve ϵ-approximate ϕ-quantiles over the last $r \leq R$ elements, the AM fixed sketch $F(2^k, \frac{\epsilon}{2})$ from $V(R, \epsilon)$ is chosen so that $2^{k-1} < r \leq 2^k$. The procedure to obtain the quantiles from that sketch is the same as above. Note that $V(R, \epsilon)$ can in principal be used to obtain the ϵ-approximate ϕ-quantiles

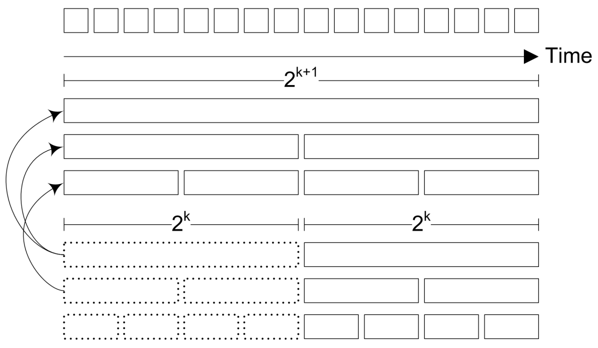


Figure 4.10: The relationship between AM fixed sketches $F(2^k, \frac{\epsilon}{2})$ and $F(2^{k+1}, \frac{\epsilon}{2})$ in an AM variable sketch $V(R, \epsilon)$. ∘

for any $r \leq R$, i.e. it is applicable to a more general class of windows called r-of-R [49]. In figure 4.10 the AM fixed sketches $F(2^k, \frac{\epsilon}{2})$ and $F(2^{k+1}, \frac{\epsilon}{2})$ in an AM variable sketch are illustrated conceptually along with how their levels are related.

In summary, one can use AM to compute ϵ-approximate ϕ-quantiles over fixed-size and variable-size retentions. Under the hood it uses GK to obtain block level quantile sketches. To compute ϵ-approximate ϕ-quantiles under an AM fixed sketch $F(R, \epsilon)$ it takes:

- Maintenance space $O(\frac{1}{\epsilon} * \log \frac{1}{\epsilon} * \log R)$ (conservative) [5], based on the maintenance space $O(\frac{1}{\epsilon_l} * \log(\epsilon_l * N_l))$ for GK on each block under construction and maintenance space $O(\frac{1}{\epsilon_l})$ for quantile sketches of active blocks.
- Update time is not mentioned in literature, however it consists of L blocks (only those under construction) running GK for the new element and potentially creating a block level sketch.
- Query time is not mentioned in literature, however it consists of determining the disjoint union of blocks, sorting the quantiles of all quantile sketches of the blocks and a sequential scan over it to obtain the quantile(s) of interest.

To compute ϵ-approximate ϕ-quantiles under an AM variable sketch $V(R, \epsilon)$ it takes:

- Maintenance space $O(\frac{1}{\epsilon} * \log \frac{1}{\epsilon} * \log R * \log \epsilon R)$ [5], to maintain at most $\lfloor \log(\epsilon R) \rfloor$ AM fixed sketches.
- Update time is not mentioned in literature, however it consists of $\lfloor \log(\epsilon R) \rfloor$ updates to AM fixed sketches.
- Query time is not mentioned in literature, however it consists of selecting an AM fixed sketch and performing a query on it.

By combining sampling with GK in AM, a randomized AM is obtained so that with $1-\delta$ probability of success the ϵ-approximate ϕ quantiles can be determined [5]. The use of intervals in AM is not limited to quantiles and can be used to determine ϵ-approximate frequency counts for heavy hitters (a most frequent items aggregate) [5]. Similar to quantiles, the frequency counts can be randomized. The idea behind AM, aggregating intervals and using a disjoint union of them for approximation, seems more generally applicable [7]. Moreover it is robust, because a relative error bound is guaranteed. The interested reader is suggested to dive into literature [5, 7].

4.8 Wrap-up

A synopsis structure is developed to represent large volumes of data (of a stream) in a space and perhaps time efficient way. They are constructed in a single pass over the data. Their performance is expressed as update time, query / lookup time and maintenance space. If a synopsis structure approximates aggregates, it should be robust, i.e. adhere to an error bound. Ideally synopsis structures are broadly applicable.

In the incremental approach a synopsis structure must have a notion of retention (hence sliding windows) and support implicit deletes. A difficulty stems from retentions that have a fixed-size, e.g.

count-based (see definition 2.3.2), or a variable-size, e.g. count-based distinct-time (see definition 2.3.3) or time-based (see definition 2.3.4). Another aspect is the support for accuracy requirements. The relative error bound ϵ categorizes synopsis structures as exact ($\epsilon = 0$) or approximate ($\epsilon > 0$). The success probability $1 - \delta$ of achieving the relative error bound categorizes synopsis structures as deterministic ($1 - \delta = 1$) or randomized / probablistic ($1 - \delta < 1$). Accuracy requirements are defined differently for variance (see definition 4.2.1) and quantiles (see definition 4.2.2). Putting it together, the following taxonomy for synopsis structures is proposed:

- Fixed number of elements (in the retention):
  - Exact ($\epsilon = 0$): expiration list (see section 4.3) and panes (see section 4.4).
  - Approximate ($\epsilon > 0$)
    * Deterministic ($1 - \delta = 1$): variance histograms (see section 4.6) and quantile summaries (see section 4.7).
    * Randomized or probablistic ($1 - \delta < 1$): sampling (see section 4.5).
- Variable number of elements (in the retention):
  - Idem.

Several synopsis structures covering the taxonomy are discussed in this chapter. An expiration list is a trivial synopsis storing all elements in the retention. Panes exploit boundedness (see definition 3.4.2) of aggregates to obtain (huge) memory space savings (in practice) compared to expiration lists when the batches are sufficiently large. However performance is similar to expiration lists when sliding 1 element at a time. Panes seem to provide hardly any benefit (if any) to quantiles. Sampling selects a subset of the elements in the retention probablistically, hence necessarily yields randomized approximations. Biased reservoir sampling is a relatively simple and efficient sampling technique compared to other techniques for sampling. Variance histograms are tailored to (ϵ-)approximate variance and are especially useful when sliding 1 element at a time. Arasu-Manku sketches are tailored to (ϵ-)approximate ϕ-quantiles. As quantiles are unbounded aggregates (see chapter 3), Arasu-Manku sketches can provide memory space savings at the expense of accuracy. The synopsis structures are used in the empirical part of the research to measure the performance of the incremental approach in practice, see chapters 5, 6 and 7. Moreover their computational complexity can be plugged into the theoretical model of 3 to compare the incremental approach using a particular synopsis structure with the instantaneous approach or with the use of another synopsis structure.

Note that many more synopsis structures are mentioned in literature [2, 10, 36] and they support various aggregates beyond variance and quantiles. The number of synopsis structures applicable to the incremental approach (and as such applicable to sliding windows) is limited. The interested reader is suggested to dive into literature pointers provided throughout this chapter.

The following **guidelines** regarding synopsis structures are derived from this chapter:

- Synopsis structures must support a notion of retention and implicit deletes, hence sliding windows, to be applicable under the incremental approach.
- Panes and the trivial synopsis structure expiration list compute exact aggregates and support both fixed-size and variable-size retentions.
- Panes exploit bounded aggregates to significantly reduce memory space usage with respect to expiration lists when the batch size is sufficiently large.
- (Biased reservoir) sampling computes randomized approximations and supports both fixed-size and variable-size retentions.
- A sample size of $O(\epsilon^{-2} * \log(\epsilon\delta) - 1)$ should suffice to compute ϵ-approximate ϕ-quantiles with success probability $1 - \delta$.
- Variance histograms compute ϵ-approximate variance over a fixed-size retention; extension to variable-size retentions is non-obvious.
- Quantile summaries compute ϵ-approximate ϕ-quantiles and support both fixed-size and variable-size retentions.

Chapter 5

Environment Setup

This chapter outlines the setup of the environment and measurements to obtain empirical results of the performance of the approaches in practice. The empirical results for the selected aggregates (see chapter 3) are presented and discussed in chapter 6. They will be used for analysis in chapter 7 to obtain empirical approximations for the theoretical model (see chapter 3).

This chapter starts with a discussion of the streaming data processing system StreamInsight used during the research in section 5.1. It discusses some interesting aspects of the system and relates some of them to the formal definitions of chapter 2. It can be skipped if one is not interested in those details. Next in section 5.2 the environment configuration is described and how processing time and memory space usage are measured on implementations of the approaches in the streaming data processing system. Additionally measuring accuracy is described, which is useful in determining qualitative performance of approximations to obtain guidelines. In section 5.4 is elaborated on the reduction of evaluation effort in the interest of time. It deals with the heuristic reduction of the number of workloads (input, aggregate, accuracy requirements, retention, slide) for approaches to obtain measurements for that are generalized by regression models in chapter 7. The chapter ends with a wrap-up in section 5.5.

5.1 StreamInsight

Several streaming data processors have been developed over time. Aurora [18], STREAM [53], NiagaraCQ [22] and TelegraphCQ [21] are just a few of the systems from academia. Recently Microsoft launched its own streaming data processor named StreamInsight, formerly known as Microsoft CEP server and based on their CEDR research [3, 4]. The aim is to deliver low-latency, high throughput continuous query processing leveraging temporal algebra and the ability to handle stream imperfections like out-of-order data. Within Info Support, a stakeholder of the research, the Business Intelligence unit uses the BI stack of products from Microsoft. The release of StreamInsight with SQL Server 2008 R2, which is used by BI, and some potential business cases arose their interest to explore StreamInsight. In the interest of time, only the streaming data processing system preferred by Info Support is considered.

In StreamInsight [3, 4, 58] the physical stream of elements is a sequence of objects (tuples) with a payload and control parameters. The control parameters contain metedata about the element, of which the interval during which an element is active plays a central role. This extends upon the objects in a stream as defined in the semantics of chapter 2 (see 2.2.2). The end time of the interval can be updated to reflect lifetime modifications of elements, e.g. setting it to the start time expresses zero lifetime also known as retraction or deletion. A special kind of element that acts as a time-based punctuation of the stream is inserted to indicate the progress of time, named current time increment (CTI) in StreamInsight. More generally, punctuations annotate the stream to specify the end of some subset of data [32, 60]; in this case regarding lifetime of elements. The timestamp t associated with the CTI indicates that no element modification of the stream will happen affecting time earlier than t. Only elements started before t are affected by the CTI. Elements whose lifetime ended before t cannot be modified, while elements started before t can have their end time updated to timestamps greater than or equal to t. By means of the CTI elements cannot be arbitrarily out-of-order. The more often a CTI is inserted, the more often elements are pushed into the processing pipeline. However, the more $CTIs$ inserted, the more elements are processed (in extreme a CTI after each element, causing the stream to double). The $CTIs$ can be used to tune and affect performance. From the physical stream a logical stream is obtained through what is called a canonical history table (CHT), providing a temporally ordered stream of elements with retractions and insertions applied and CTI elements removed. Note how this matches the definition of the semantics of a (logical) stream (see 2.2.2) with time monotonicity (see 2.2.3).

Data comes into the streaming data processing system by means of input adapters and comes out through output adapters. An operator that processes data can consume a stream of input and produces a stream of output. A tree of operators forms a query that can be expressed using LINQ, a declarative query language integrated into .NET languages with compile-time support. An example LINQ query in C# that computes the average over a window is shown in algorithm 1. Queries are fed by input adapters and stream their results to output adapters. The adapters can be implemented in .NET and provide great freedom in capturing stream sources and producing output [52]. Queries can be declared by query templates of which multiple instances can be executed. Some optimization is happening behind the scenes [3]. Each operator in an operator tree of a query is represented as a task in the runtime. Operators can be optimized by measuring selectivity and throughput to guide cost-based query optimization. Tasks representing operators can exchange data, be assigned to CPU cores and can cross streaming data processing instance and machine boundaries (operator migration). This model allows distributed parallel query execution. To manage task overhead, horizontal and vertical fusion is employed. Vertical fusion causes two operators to be executed in a single task, e.g. when a filter operator runs on top of another operator they can be fused. Horizontal fusion maps identical operators from different query instances with the same template into the same task. Paradoxically, another optimization is partitioning to divide heavy workload into smaller workloads. Streams can be partitioned using the a group-and-apply approach that clones a query into multiple queries and applies each query to a partition of the stream. Queries can be partitioned by dividing a query into subqueries that can be deployed on different instances of the streaming data processing system spanning machine boundaries.

Algorithm 1: An example LINQ query in C# for StreamInsight computing the average over a time-based sliding window with a retention (and in this case extent) of 1 hour and a slide of 1 minute using the built-in aggregate. Details on query formulation are in [52].

```
**from** $window$ **in**
$input.HoppingWindow(TimeSpan.FromHours(1), TimeSpan.FromMinutes(1),$
$HoppingWindowOutputPolicy.ClipToWindowEnd)$
**select new** { $SomeAverage = window.Avg(e => e.Value)$ };
```

There are several kinds of operators defined for StreamInsight [3, 4, 52, 58]. Span-operators perform computations over each input element and produce an output elements whose lifetime or span is the input element lifetime for single-input operators like projection and filtering, while it is the intersection of input lifetimes for multi-input operators like (temporal) join. The multicast operator is used to output a single stream as multiple similar streams to multiple operators. The union operator is used to merge multiple streams into a single stream. These operators are related to the grouping of a stream, i.e. partitioning it in a sub stream per group, applying some branch of the operator tree to each sub stream in parallel and merging the results into a single stream. The window-based operators compute results based on the elements in the window. The sliding windows are count-based distinct-time or time-based. Their semantics is similar to definitions 2.3.3 and 2.3.4. A limitation of StreamInsight is that count-based distinct-time windows are only able to slide one distinct-start time (rather than a variable number) and no support exists for count-based windows (see definition 2.3.2). Count-based windows with variable slide can be simulated by a stream with a fixed rate of elements (no duplicates) and time-based windows. Note that such a stream might not be available in a practical setting, e.g. due to bursty element rates.

Next to windows, a limited number of built-in aggregates are available with no support for count-based windows [52]: min, max, sum, count, average and top-k. A short remark on the built-in aggregates of StreamInsight has to be made. While they are potentially the most optimized aggregates for StreamInsight, they unfortunately cannot be used in the research. From a development point of view they are black boxes with their implementations hidden inside StreamInsight. Consequently they cannot be measured like the user defined aggregate implementations. Moreover the approach followed by their implementations is not known. The built-in aggregates are excluded from the research. StreamInsight can be extended with user-defined functions (UDFs), user-defined aggregates (UDAs) and user-defind operators (UDOs) implemented in .NET [4, 52], collectively referred to as user-defined modules (UDMs). UDFs effectively are method calls, with arbitrary parameters and a single return value. They can be used wherever span-based operators (expressions) occur, like filter and join predicates. UDAs and UDOs are operators that aggregate over the elements of a window. To work around the optimization boundary of

Environment configuration	
Processor	3 GHz dual core Intel Pentium D
Main memory	8 GB
Operating System	Microsoft Windows 7 Enterprise 64-bits
Virtualization	None
Runtime	.NET 4.0 64-bits
Programming Language	C#
Streaming Data Processing System	Microsoft StreamInsight v1.1
Stream	Uniform random double-precision floating points
Element rate	± 320 elements per second (a CTI every 20 elements)

Table 5.1: Configuration of the environment used during measurements. ◦

a UDM, which is a black box to query optimization, two approaches seem fruitful [4]. The first approach requires that the UDM specifies properties, e.g. selectivity or expected CPU load per input tuple, to guide the cost-based query optimizer. The second approach requires the streaming data processing system to measure properties of the operators itself and using the results to guide optimization. The latter benefits UDM development by releasing the burden of specifying the properties (correctly), but suffers from suboptimal optimization and performance during a learning period for the properties. Nevertheless, the latter approach seems to be employed currently.

5.2 Environment

In the interest of time a fixed environment with a single streaming data processing system preferred by Info Support will be used. The streaming data processing system for which the aggregates are implemented is Microsoft StreamInsight version 1.1, which at the time of writing and evaluation is the most recent stable version. The input and output adapters, operators and query are all expressed in a .NET language, of which C# is used during the research. This is a preference, but should not impact the results. In .NET there is a common language runtime (CLR) that executes an intermediate language, analogous to Java bytecode. Languages like Basic and C# are compiled into the intermediate language. The CLR is a managed environment, i.e. memory management is a burden of the runtime by means of a garbage collector rather than a burden of the programmer. However, the garbage collector is concurrent and runs non-deterministically. The environment configuration is summarized in table 5.2.

In absence of real data, uniform random numbers were generated to produce a stream of synthetic data. A stable rate of about 320 elements per second (a CTI every 20 elements) is achieved in the environment, yielding slightly less than 20000 elements per minute. Count-based retentions can be simulated this way by time-based retentions that are better supported in StreamInsight. Furthermore aggregates are computed over (approximately) the same number of elements every time. Moreover the rate allows reasonable sized retentions in a reasonable amount of time, e.g. well over a million of events within an hour. Attempts to increase the rate caused it to become unstable; further optimization seems possible though absence of need and time prevented investigation thereof.

The implementations of the approaches are user defined aggregates (UDAs) which become operators in a query. A user defined aggregate consists of two parts: a class of the type CepTimeSensitiveAggregate and an extension method that invokes the aggregate for use in queries [52]. The CepTimeSensitiveAggregate is a base type for aggregates that require both the payload and control parameters of each event in the window when computing the aggregate (if only payload is needed the type CepAggregate suffices). The implementation is straightforward since each time the streaming data processing system has a window of data, the GenerateOutput method of the aggregate is invoked passing an enumerator over the window. Note that by design, every call to the GenerateOutput method must produce an aggregate. *This forces the batch / small window size to be equal to the slide!* Otherwise the incremental approach produces output more frequent than the instantaneous approach violating the output semantics of the approaches. As a consequence, the performance of larger slides and smaller batches cannot be observed in the proposed setup. Looking at the theoretical model of chapter 3 this most likely affects memory space usage $S(B, R)$ of the incremental approach, because it depends on the batch size and synopsis structure covering the retention. Regarding processing time $T(R, S, B) = S * T_u(R) + T_q(R)$, the choice of batch size seems insignificant, although it does affect per batch processing time. The internal state by means of some synopsis structure can be kept in instance variables of the user defined aggregate (spanning the lifetime of an operator in a query). Queries using the user defined aggregate as an operator should be as simple as possible, e.g. only perform the aggregate under the approach represented by the operator, to isolate the performance of the operator and be able to measure it. In figure 5.1 a query with input and output adapters is schematically depicted. More details are described next.

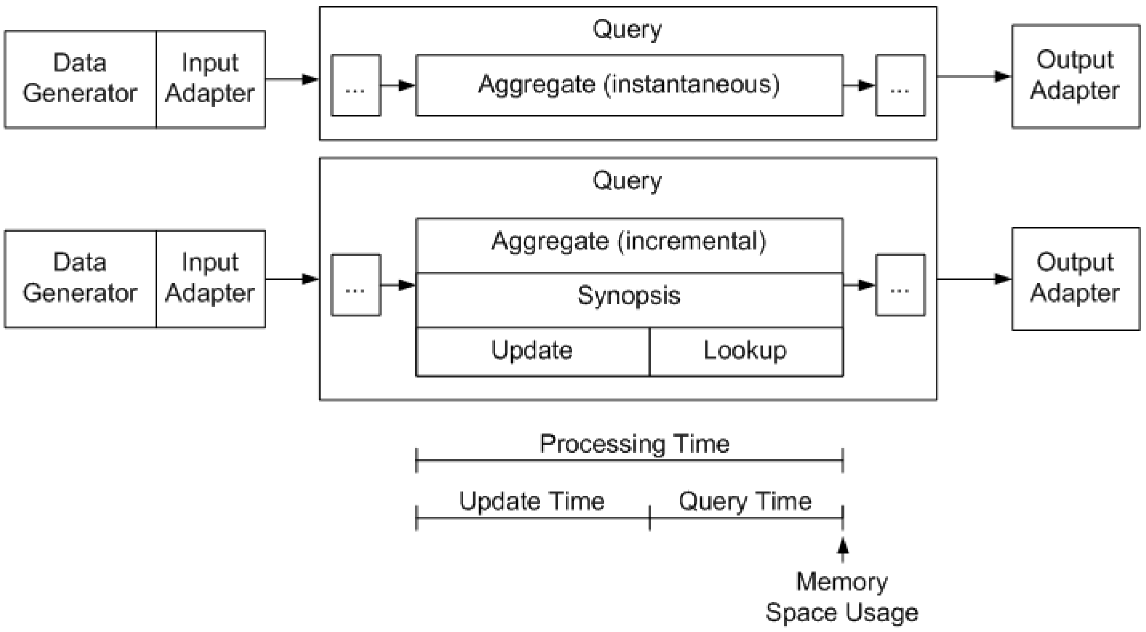


Figure 5.1: An input adapter has elements with data from the synthetic data generator. This is fed to the query running the aggregate operator (and some default operators). The aggregate results end up in an output adapter. The bottom query runs the implementation of an aggregate with a synopsis structure under the incremental approach. Processing time and memory space usage are measured for the aggregate operator as indicated. The top query represents an aggregate under the instantaneous approach. ○

5.3 Measures

Regarding performance processing time and memory space usage are measured. To obtain guidelines on accuracy of synopsis structures producing approximated aggregates, exact and approximated output are compared.The measurements make up the empirical part of the research and are presented in chapter 6. They are used for analysis described in chapter 7 to obtain empirical approximations that can be plugged into the theoretical model of chapter 3. Below the process of measuring is explained and the measures are defined. It is depicted schematically in figure 5.1.

The computation in the user defined aggregate (operator) can be broken down into the following phases:

1. Prolog: just before the aggregate starts to do its actual work.
2. Body: the aggregate is computed. For aggregates under the incremental approach:
   (a) Update: update the synopsis structure with new elements (and account for expiration).
   (b) Query / Lookup: retrieve the aggregate from or based on the synopsis structure.
3. Epilog: just before the aggregate output is returned.

In figure A.1 in the appendix a C# code template is shown indicating the phases, which is used for all aggregate implementations for this research. The details of the measures are discussed next.

Definition 5.3.1 (Processing Time). *Processing time is the actual number of seconds between the prolog and epilog to compute an aggregate over the retention; the empirical interpretation of $T(R)$ of the instantaneous approach and $T(R, S, B)$ of the incremental approach in the theoretical model (see chapter 3). The update time comprises $S * T_u(R)$ and query time comprises $T_q(R)$.*

For the instantaneous approach processing time measurement happens per large window, for the incremental approach measurement happens per slide. The maintenance time is the time between the prolog and just before the look up. It represents the time to update the synopsis structure. The query / lookup time is the time after updating until the epilog. It represents the time to determine the aggregate from the synopsis structure. Note that such a division of processing time in maintenance and query time is non-obvious for the instantaneous approach due to the absence of a synopsis structure, hence it is omitted for the instantaneous approach. Processing time is achieved by stopwatch functionality, e.g. the Stopwatch type in the base class library of .NET. The high-resolution stopwatch provides reasonably accurate results. It measures the ticks (100-nanosecond intervals) elapsed between start and end of the stopwatch and provides the number of ticks per second. The stopwatch is assumed to have insignificant overhead. The second to millisecond resolution seems to suffice in comparing the approaches without being susceptible to minor fluctuations in the environment. The processing, maintenance and query time measurements are indicated in figure 5.1.

Definition 5.3.2 (Memory Space Usage)**.** *Memory space usage is the actual amount of bytes the streaming data processing system uses when computing aggregates over the retention; the empirical interpretation of $S(R)$ of the instantaneous approach and $S(B, R)$ of the incremental approach in the theoretical model (see chapter 3).*

For the instantaneous approach memory space usage measurement happens per large window, for the incremental approach measurement happens per slide. The memory space usage is measured just before the epilog, after the synopsis is updated and queried. At that point in time the synopsis structure represents the latest state and reflect the current retention. Temporary memory space fluctuations during synopsis structure updating or querying might not show up (the garbage collector might have run and released memory already). Measuring at multiple points can cause measurements to suffer from autocorrelation: measurements closer in time are more alike than measurements more distant in time; measurements are not independent by design in that case, undermining randomness. This is unfavorable for analysis of the measurements as explained in chapter 7.

Memory space usage is measured in terms of: process private bytes and CLR heap bytes, both exposed as operating system performance counters. The process private bytes represents the number of bytes the process does not share, including the CLR heap. It is assumed it captures the large windows or batches / small windows, synopsis structure and overhead. The CLR heap is governed by the .NET runtime and is measured by the amount of maximum bytes for allocation. The managed heap consists of a small and large object heap and is logically segmented in garbage collection generations. It captures the variable upper bound of memory space usage of .NET (adjusting its needs) and is garbage collected. It is assumed that the memory space allocated by the user defined aggregate is captured by the CLR heap plus some overhead of the .NET runtime. This way a rough estimate of memory space usage is obtained in computing the aggregate under the approaches, e.g. the synopsis structure and components of the StreamInsight streaming data processing system like windows. Note that in contrast with processing time, memory space usage is coarse grained. Measuring the exact size of objects in the CLR requires dumps of the heap to be taken and analyzed. A tedious and intrusive process in the absence of automation and the processing overhead caused by such dumps. Moreover, the closed nature of StreamInsight prevents detailed insight into the memory space usage of its components. The performance counters are assumed to have insignificant overhead. The megabyte to kilobyte resolution seems to suffice in comparing the approaches without being susceptible to minor fluctuations in the environment. The memory space usage measurement is indicated in figure 5.1.

By measuring the accuracy of the approximated aggregates, guidelines can be derived regarding the error bound and success probability of achieving the error bound. It can provide insight into the qualitative performance of approximations and the synopsis structures responsible for it. The accuracy requirements are described by relative error bound ϵ and success probability $1 - \delta$. From measurements they can be determined as observed relative error $\hat{\epsilon}$ and observed success probability $1 - \hat{\delta}$.

Definition 5.3.3 (Observed Relative Error)**.** *The observed relative error $\hat{\epsilon}$ is:*

$$\hat{\epsilon} = \frac{|exact\ output - approximated\ output|}{exact\ output}$$

As the relative error is nearer to 0, the approximation is closer to the exact answer and vice versa.

In randomized / probablistic algorithms the notion of success probability pops up, e.g. sampling. The probability of success that a given relative error bound is achieved is $1 - \delta$, where δ ranges from 0 to 1.

Definition 5.3.4 (Observed Success Probability). *The observed success probability* $1 - \hat{\delta}$ *is:*

$$1 - \hat{\delta} = 1 - \frac{\textit{the number of approximations respecting } \epsilon}{\textit{the number of approximations}}$$

A lower value for δ *means a higher success probability of achieving the relative error bound* ϵ *and vice versa.*

Measuring accuracy requires both the approximated and exact aggregate. This is achieved by running an aggregate under the instantaneous approach for the exact aggregate computation and the same aggregate under the incremental approach for the approximated aggregate computation over the same input data. The parallel processing of the approaches correlates their processing processing time and memory space usage. As such it seems inappropriate to measure processing time, memory space usage and (observed) accuracy at the same time. Processing time and memory space usage are measured in isolation and (observed) accuracy is measured in isolation. Note that measuring accuracy need not necessarily happen in StreamInsight by means of queries / aggregate operators. Accuracy is not determined by technological aspects or performance influences of the streaming data processing system, but by the algorithms (e.g. maintaining synopsis structures). Hence accuracy can be measured by running the algorithms on a stream of data with the same kind of data, at the same rate used during performance measurements and with similar use of retention and slide. In terms of StreamInsight it would require running a query for an instantaneous aggregate and one for a corresponding incremental aggregate. Their outputs must then be compared. Accuracy measurement is schematically shown in figure 5.2.

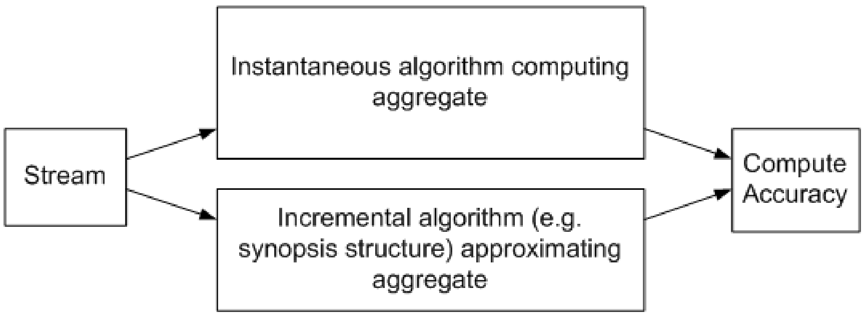

Instantaneous algorithm computing aggregate
Stream
Compute Accuracy
Incremental algorithm (e.g. synopsis structure) approximating aggregate

Figure 5.2: Accuracy is measured by taking the output from an instantaneous algorithm and from an incremental algorithm (e.g. synopsis structure) on the same data of the stream (e.g. same number of elements in the retention). Their outputs can be compared to determine the accuracy. This is way accuracy is measured. ◦

Measurements are output upon every aggregate computation. It requires some time for the streaming data processing system to start up and run in a steady-state. More precise, results can be skewed as the streaming data processing initializes processing and no full retention can be processed yet. For example a retention of 1 hour requires the query to run for at least 1 hour before measurements can be considered useful, i.e. before a steady state is achieved. The results before the steady state are discarded.

5.4 Evaluation Reduction

Performing a full empirical evaluation is beyond the scope of this research in particular because of the limited time. This section heuristically reduces the evaluation effort required. As stated before the streaming data processing system is limited to StreamInsight, which is of most interest to Info Support. This implies a Windows operating system and .NET 3.5/4.0 runtime and programming languages (of which C# is used). The environment is constrained this way and fixed during evaluation.

During evaluation a workload represents particular output semantics (see chapter 1) characterized by input, aggregate, retention, slide and accuracy. The input is fixed to uniform random numbers in absence of non-synthetic data. It is setup to produce a stable rate of about 320 elements per second. In chapter 3 the aggregates are limited those of (most) interest to Info Support: variance and quantiles. Regarding Info Support (and practical business cases in general) it seems unlikely relative errors exceeding 5% are acceptable, constraining choices of ϵ. A quick look at the complexities reported for the synopsis structures, reveals the influence of the relative error bound ϵ. Note that success probability $1 - \delta$ plays

$\delta \backslash \epsilon$	0.1	0.05	0.01
0.01	997	4386	132877
0.001	1329	5715	166096
0.0001	1661	7044	199316

Table 5.2: Sample sizes for $O(\frac{1}{\epsilon^2} * \log \frac{1}{\epsilon * \delta})$ with different success probabilities $1 - \delta$ and relative errors ϵ. $\circ$

Retention (minutes)	Slide (minutes)	Effect
1	1	Base-level overhead.
15	1	Increasing amount of data over which is aggregated.
60	1	Increasing amount of data over which is aggregated.
60	15	Decreasing overlap between consecutive aggregates.
60	30	Decreasing overlap between consecutive aggregates.

Table 5.3: Workload retention and slide combinations with effects. $\circ$

a minor role and applies only to sampling. Suppose for example the sample size $O(\frac{1}{\epsilon^2} * \log \frac{1}{\epsilon * \delta})$ and $\epsilon \in \{0.1, 0.05, 0.01\}$, $\delta \in \{0.01, 0.001, 0.0001\}$. The sample sizes are shown in table 5.4. The relative error bound has a major effect on the sample size, as opposed to success probability. Moreover, relative errors of 0.1 and 0.05 are close with respect to sample sizes, while 0.01 causes a sudden increase. Hence the relative error bounds are limited to $\epsilon = 0.05$ and $\epsilon = 0.01$ (next to exact computations) and the success probability is fixed to $\delta = 0.0001$ and. Similar arguments hold for variance histograms and quantile summaries too. Note that accuracy is not related to the choice of slide, but only to retention. Slide only affects elements covered by consecutive aggregate computations. Note that approximations require an additional accuracy evaluation (under varying retention only) as argued earlier this chapter.

For the workload to be measurable, it must exceed the base level overhead (noise) of the environment. This is overcome by making sure the workloads affect memory space usage in the megabyte to kilobyte range so that small (e.g. > 1 KB) memory changes can be ignored as noise. Similarly workloads must affect processing time in the second to millisecond range so that small (e.g. > 1 ms) time changes can be ignored as noise. Clearly the retention must cover a relatively large amount of elements to have a measurable impact on computation. Additionally it must govern slides of several magnitudes so that slides have a measurable impact too. Achieving a rate of about 320 elements per second provides a little less than 20000 elements per minute and over 1 million per hour. Assuming 64-bit double precision floating point values make up the payload of the elements plus additional overhead of objects representing such an element, the megabyte range should be achievable. If it takes time in the microsecond range get such an object (e.g. from iteration over the window), the millisecond range should be achievable.

Under a fixed slide and varying retention, the impact of retention on performance is assessed. Under a varying slide and fixed retention, the impact of slide and hence overlap on performance is assessed. Effectively retention and slide are evaluated orthogonally. Note that a retention of 1 hour and a slide of 30 minutes produces only 2 measurements per hour, requiring over a day to obtain 50 measurements. Hence for practical purposes the retention is bounded above by 1 hour and the slide by 30 minutes; a large retention and large slide respectively. Additionally a slide of 15 minutes is proposed to capture a moderate slide and a slide of 1 minute captures a small slide. A retention of 15 minutes is proposed to capture a moderate retention and a retention of 1 minute captures a small retention. The retention and slide combinations are shown in table 5.4. Apart from the practical upper bound, the choice for retention and slide is somewhat arbitrary. Most importantly, the retention changes by several magnitudes and slide analogously. Furthermore it is important that the effect of retention is evaluated by more than two combinations to reduce the chance of only apparent linear changes in performance from being observed; analogously for the effect of slide. Thus the somewhat arbitrary choice is not considered a big issue.

Workloads are evaluated under the instantaneous and incremental approaches. The latter is differentiated by the synopsis structure that is used. In the context of the selected aggregates variance and quantiles (see chapter 3) a few synopsis structures were highlighted (see chapter 4). The highlighted synopsis structures cover the proposed taxonomy, hence the incremental approach is differentiated by: expiration lists, panes, biased reservoir sampling, variance histograms and quantile summaries. As argued in chapter 4 quantiles seem not to benefit in any way from panes, so it is used only for variance. Additionally it is argued in chapter 4 that biased reservoir sampling seems very performance efficient with respect to other sampling techniques, hence it is preferred.

Following the above reasoning, evaluation effort is reduced heuristically, refraining from a full empirical evaluation. Anyone interested is free to indulge in a more extensive empirical evaluation. Note that in the next chapter 7, regression models are derived from the measurement results. They generalize the results on the environment, providing another argument for evaluation effort reduction. An overview of the empirical evaluation is shown in table 5.4.

5.5 Wrap-up

This chapter discussed the setup of empirical measurements of performance of the approaches whose results are discussed in chapter 6. The empirical measurements of performance are used in chapter 7 to obtain empirical approximations that can be plugged into the theoretical model of chapter 3. The discussion is started with the streaming data processing system Microsoft StreamInsight v1.1. Some parts in the system can be linked to semantics described in chapter 2, like (logical) streams and windows. A uniform random number generator will be used to generate elements of a stream, in absence of non-synthetic data. Count-based windows are not supported, but can be simulated using time-based windows and a stream with a fixed rate of elements. An additional advantage is that every time an aggregate is computed, it is computed over the same amount of data, i.e. each retention has (about) the same number of elements.

The approaches are implemented as user defined aggregates (UDAs) which become operators in queries on the system. For the incremental approach the batch size must be equal to the slide in the system, as by design a UDA needs to produce an aggregate every time it is invoked. Output semantics would be violated if batch sizes are smaller than the slide on the system, as upon every batch an aggregate would be produced rather than upon every slide. In a sense the distinction between batches and slides cannot be implemented on StreamInsight.

The actual time from just before the operator starts computing the aggregate until just before it returns the computed aggregate is defined as processing time; $T(R)$ for the instantaneous approach and $T(R, S, B)$ for the incremental approach. It is measured by stopwatch functionality in the second to millisecond resolution to prevent minor fluctuations from skewing the measurements. For the incremental approach processing time is split out in update time $S*T_u(R)$ and query / lookup time $T_q(R)$. The actual amount of memory used by the streaming data processing system to compute the aggregate is defined as memory space usage; $S(R)$ for the instantaneous approach and $S(B, R)$ for the incremental approach. It is measured by operating system performance counters in the megabyte to kilobyte resolution to prevent minor fluctuations from skewing the measurements. To obtain guidelines on the qualitative performance of approximations, observed accuracy is measured in terms of observed relative error and observed success probability of achieving the relative error. This requires the output of the instantaneous approach to be compared to output of the incremental approach. As the approaches run simultaneously, their processing time and memory space usage correlate which is inappropriate. Processing time and memory space usage is measured in isolation and observed accuracy is measured in isolation. The measurement setup is schematically depicted in figure 5.1.

In the last part of the chapter it is argued that a full empirical evaluation is infeasible and hence is reduced heuristically. First of all the environment is fixed and streams consist only of uniform random numbers at a rate of about 320 elements per second. Only a limited selection of aggregates is evaluated (the aggregates were selected in chapter 3). Only a limited number of synopsis structures is used (they are described in chapter 4) for the incremental approach. It is argued that a small range of relative errors is acceptable in a practical setting and success probability plays a minor role, hence $\epsilon \in \{0.05, 0.01, 0\}$ are used with a fixed success probability ($\delta = 0.0001$). Moreover not all combinations of retention and slide are evaluated. To see the effect of retention on performance, a variable retention (1, 15 and 60 minutes) and fixed slide (1 minute) is used. To see the effect of slide (or inversely: overlap), a fixed retention (60 minutes) and variable slide (1, 15 and 30 minutes) is used. StreamInsight forces user defined aggregates to output for every window. However, the incremental approach has small windows / batches that of which several make up a slide. This can cause the incremental approach to output more frequently than the instantaneous approach, violating output semantics. To circumvent this issue, batch sizes are constrained to be equal to the slide. This also constrains the retention to be in integer multiple of the slide, and hence batch size, to satisfy the constraints mentioned in chapter 2). An overview of the evaluations is given in table 5.4.

Retention	**Slide**	**Approach**		ϵ	**Aggregate**
1	1	Instantaneous		0	V, Q
		Incremental	Expiration List	0	V,Q
			Panes	0	V
			Sampling	0.01	V,Q
				0.05	V,Q
			Histogram	0.01	V
				0.05	V
			Quantile Summary	0.01	Q
				0.05	Q
15	1	Instantaneous		0	V, Q
		Incremental	Expiration List	0	V,Q
			Panes	0	V
			Sampling	0.01	V,Q
				0.05	V,Q
			Histogram	0.01	V
				0.05	V
			Quantile Summary	0.01	Q
				0.05	Q
60	1	Instantaneous		0	V, Q
		Incremental	Expiration List	0	V,Q
			Panes	0	V
			Sampling	0.01	V,Q
				0.05	V,Q
			Histogram	0.01	V
				0.05	V
			Quantile Summary	0.01	Q
				0.05	Q
	15	Instantaneous		0	V, Q
		Incremental	Expiration List	0	V,Q
			Panes	0	V
			Sampling	0.01	V,Q
				0.05	V,Q
			Histogram	0.01	V
				0.05	V
			Quantile Summary	0.01	Q
				0.05	Q
	30	Instantaneous		0	V, Q
		Incremental	Expiration List	0	V,Q
			Panes	0	V
			Sampling	0.01	V,Q
				0.05	V,Q
			Histogram	0.01	V
				0.05	V
			Quantile Summary	0.01	Q
				0.05	Q

Table 5.4: An overview of the workloads making up the empirical evaluation effort. The bounded differential aggregate variance is denoted V and unbounded non-differential aggregate quantiles is denoted Q. For the retention periods of 1, 15 and 60 minutes the approaches with relative error bound > 0, require an additional accuracy evaluation. For example, quantiles are evaluated with a retention of 60 minutes and a slide of 15 minutes under an incremental approach using sampling with relative error bounds of 0.05 and 0.01 and success probability 0.0001; this is a workload. ◦

Chapter 6

Empirical Results

This chapter presents the empirical results from measurements of performance in terms of actual processing time and memory space usage under workloads and approaches on StreamInsight v1.1 described in chapter 5 and summarized in table 5.4.

The processing time and memory space usage results for the bounded differential aggregate variance are discussed in section 6.2.1. The (observed) accuracy results of approximations of variance follow in section 6.2.2. The processing time and memory space usage results for the unbounded non-differential aggregate quantiles are discussed in section 6.3.1. The observed accuracy results of approximations of quantiles follow in section 6.3.2. The chapter ends with a wrap-up in section 6.4.

6.1 Preliminary

In terms of the theoretical model of chapter 3 the reported operator time resembles the processing time $T(R)$ for the instantaneous approach and $T(R, S, B)$ for the incremental approach. The reported update time resembles $S * T_u(R)$. By the fixed element rate one can obtain $T_u(R)$. The reported query time resembles $T_q(R)$. The reported process private memory resembles $S(R)$ for the instantaneous approach and $S(B, R)$ for the incremental approach. The reported CLR heap is the .NET part of the memory space usage, which is used by the implementations for storage. Note that during the measurements the batch size B and slide S were equal. This is caused by user defined aggregates in StreamInsight requiring output for every window. However, multiple batches / small windows in the incremental approach can make up a slide and consequently output more frequently regarding the instantaneous approach. This would break output semantics. An equal batch size and slide circumvents it. Details are in chapter 5. Looking at the theoretical model, this most likely affects $S(B, R)$ since memory space usage is both dependent on the batch size and synopsis structure covering the retention. From $T(R, S, B) = S * T_u(R) + T_q(R)$ the choice of batch size seems insignificant. Clearly, as batch sizes are smaller, memory space usage and per batch processing time reduces. In general batch sizes are more or less independent of the slide (except for some constraints mentioned in chapter 2). This means that even if a large slide is used, small batches can be employed. Unfortunately, this cannot be observed in the current setup where the batch size is equal to the slide.

Measuring is performed for at least 8 hours to produce at least 50 usable measurements per workload. The initial hour of measurements was removed, assuming that a retention of up to 60 minutes has the streaming data processing system in a steady state after the first 60 minutes. More measurements are taken when the slide is small and vice versa. Note that obtaining 50 usable measurements on a retention of 60 minutes and slide of 30 minutes takes over day! The results are randomly subsampled into datasets of 50 measurements. This way each workload is equally well represented by those datasets. The datasets are used in the next chapter on regression analysis 7 to obtain regression models. The (observed) accuracy evaluations were performed separately from the those measuring processing time and memory space usage as discussed in chapter 5. Since accuracy relates to the retention, slide plays no role. No subsampling is performed as no further analysis will be performed on the accuracy results. Their purpose is to demonstrate observed accuracy to obtain guidelines on approximations.

6.2 Results: Variance

This section shows the empirical results for the bounded differential aggregate variance summarized in graphs and tables. A discussion of the results in text follows. The quantitative performance in terms of actual processing time and memory space usage is treated in section 6.2.1. The qualitative performance in terms of observed accuracy is treated in section 6.2.2.

6.2.1 Performance

The aggregate variance is evaluated under the instantaneous approach; see results in table 6.1. Furthermore it is evaluated under the incremental approach by means of biased reservoir sampling (see section 4.5, expiration lists (see section 4.3), panes (see section 4.4) and variance histograms (see section 4.6). See tables 6.2 and 6.3 for results on biased reservoir sampling; see table 6.4 for results on expiration lists; see table 6.5 for results on panes; see tables 6.6 and 6.7 for results on variance histograms. The relationships of processing time versus retention and slide are shown in the graphs 6.1 and 6.2. Similarly relationships of memory space usage versus retention and slide are shown in the graphs 6.3 and 6.4.

The instantaneous approach (see table 6.1 and graphs 6.1, 6.2, 6.3 and 6.4) shows both processing time and memory space usage grow linearly as the retention increases. However, processing time and memory space usage are unaffected by changes in the slide. This confirms $T(R)$ and $S(R)$ from the theoretical model in chapter 3 that are dependent only on the number of elements in the retention R. Moreover it is in line with computational complexity mentioned in proposition 3.2.4. The CLR heap is hardly utilized and remains constant as either retention or slide increases. Likely the large window of data is kept outside the .NET runtime and hence outside the CLR heap. The standard deviation of the measurements is low, suggesting a rather stable usage of system resources.

The incremental approach using biased reservoir sampling with $\delta = 0.0001$ perform similar for $\epsilon = 0.05$ and $\epsilon = 0.01$ (see table 6.2 and 6.3 and graphs 6.1, 6.2, 6.3 and 6.4). Increased accuracy comes at a small cost. They show a near constant amount of memory space usage and processing time as the retention increases, while growing linearly as the slide and hence batch size increase. The constant amount of memory space usage as the retention increases can be explained by the choice of the sample size independent of the retention, $O(\epsilon^{-2}{*}\log(\epsilon\delta)^{-1})$. Clearly as the batch size increases, the memory space usage increases too as more elements are kept in memory. Processing time is constant as the retention increases, because the batch size is fixed and under the incremental approach processing happens per batch. Obviously as the batch size increases, so does the processing time as more elements need to be processed. At about a slide of half the retention, $S \approx \frac{1}{2} * R$, the overhead of sampling is $S * T_u(R)$ and exceeds the processing time of the instantaneous approach $T(R)$. Note that query time $T_q(R)$ is negligible as confirmed by the results; it is no more than retrieving the internally maintained variance. Up to $S \approx \frac{1}{2} * R$ there is still a clear memory space usage benefit by $S(B, R)$ with respect to memory space usage of the instantaneous approach $S(R)$. The standard deviation of the measurements is low, suggesting a rather stable usage of system resources.

The incremental approach using expiration lists (see table 6.4 and graphs 6.1, 6.2, 6.3 and 6.4) shows a near constant processing time and near linear growth of memory space usage as the retention increases. The constant processing time is explained by the fixed batch size. The linearly growing memory space usage is explained by the triviality of the synopsis structure: storing all elements in the retention similar to a large window. Processing time and memory space usage both grow linearly as the slide and batch size increase. This is caused by batches increasing in size. From the large amount of memory of the CLR heap, it seems the buckets of the expiration list are stored on it. At about a slide $S \approx \frac{1}{3} * R$, the overhead of maintaining an expiration list is $S * T_u(R)$ and exceeds the processing time of the instantaneous approach $T(R)$. Again query time is negligible. There is no memory space usage benefit by $S(B, R)$ with respect to the memory space usage of the instantaneous approach $S(R)$. It seems a bucket requires more memory space than a window element.

The incremental approach using panes (see table 6.5 and graphs 6.1, 6.2, 6.3 and 6.4) shows performance similar to sampling. The near constant processing time and memory space usage as the retention increases can be explained by the small number of panes. At most 60 panes are required for a retention of 60 minutes and a slide and batch of 1 minute. Likely similar results could be obtained for other bounded aggregates, like average and even minimum / maximum, on such a small number of panes. As the slide increases so does the batch size. Linearly growing processing time and memory space usage are observed as more elements are processed and kept in memory. At about a slide $S \approx \frac{1}{2} * R$, the overhead of maintaining panes is $S * T_u(R)$ and exceeds the processing time of the instantaneous approach $T(R)$. Up to $S \approx \frac{1}{2} * R$ there is still a clear memory space usage benefit by $S(B, R)$ with respect to memory space usage of the instantaneous approach $S(R)$. This is the benefit of the small number of panes. A benefit lost as the slide becomes sufficiently small and / or the retention becomes sufficiently large, causing the number of panes to increase. The standard deviation of the measurements is low, suggesting a rather stable usage of system resources.

The incremental approach using variance histograms with $\epsilon = 0.05$ (see table 6.6 and 6.7 and graphs 6.1, 6.2, 6.3 and 6.4) shows near constant processing time and memory space usage as the retention increases. This is explained by the small slide and batch size and the histograms storing a relatively small number of buckets. The processing time and memory space usage grow linearly as the slide and batch size increase, because more elements need to be processed and kept in memory. At about a slide a quarter of the retention, $S \approx \frac{1}{4} * R$, the overhead of maintaining the histogram is $S * T_u(R)$ and exceeds the processing time of the instantaneous approach $T(R)$. At about a slide $S \approx \frac{2}{5} * R$, the memory space usage $S(B, R)$ exceeds the memory space usage of the instantaneous approach $S(R)$. The incremental approach using variance histograms with $\epsilon = 0.01$ (see table 6.6 and 6.3 and graphs 6.1, 6.2, 6.3 and 6.4) shows a logarithmically growing processing time and memory space usage as the retention increases. This can be explained by the computational complexity of variance histograms being logarithmic in the retention. Processing time grows linearly as the slide and batch size increase. Memory space usage shows an anomalous pattern. This could not be explained. At a slide $S \approx \frac{1}{8} * R$, the overhead of maintaining the high accuracy variance histograms is $S * T_u(R)$ and exceeds the processing time of the instantaneous approach $T(R)$. The query time is negligible for variance histograms; it is a slightly more involved computation than simple retrieval of an internally maintained aggregate, but can be performed in quickly (computationally in constant time). There is hardly a memory space usage benefit at small slides and batch sizes. The logarithmic growth of memory space usage will become more beneficial as the retention increases sufficiently. Note that the relative error can cause a significant multiplicative factor regarding computational complexity; in the order of $\frac{1}{\epsilon^2}$ (see chapter 4 regarding histograms in section 4.6).

In general it can be observed that the instantaneous approach shows performance growth linear in the size of the retention (as expected) and independent of the slide. The incremental approach necessarily has performance grow as the slide and batch size increase, because larger batches with more elements are processed. Sampling and panes perform best. However, sampling is a probablistic / randomized approximation. The performance of panes depends heavily on the number of panes representing the retention. Variance histograms can do a good job at reducing memory space usage and improving upon processing time as well with respect to the instantaneous approach. Another benefit is its deterministic behavior as opposed to sampling. The expiration list has only processing time benefits as the slide is relatively small. In general processing time of the incremental approach has lost its benefit at about a slide of half the retention. This is caused by update time $S * T_u(R)$ as the overhead of a query / lookup $T_q(R)$ is negligible. Memory space benefits are possible beyond a slide of half the retention, e.g. by sampling and panes. It should be kept in mind that during evaluation only uniform random values were aggregated over. Other distributions can yield different results, especially if they are favored or disfavored by certain algorithms.

Instantaneous, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.008	0.000	0.126	0.002	0.473	0.007
Process Private Memory	(MB)	26.974	0.008	46.029	0.008	99.122	0.009
CLR Heap	(MB)	1.289	0.043	1.260	0.007	1.256	0.002

Instantaneous, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.473	0.007	0.493	0.017	0.497	0.010
Process Private Memory	(MB)	99.122	0.009	98.561	0.008	101.423	0.136
CLR Heap	(MB)	1.256	0.002	1.257	0.009	1.258	0.013

Table 6.1: Measurements for computing the bounded differential aggregate variance under the instantaneous approach with a single-pass incremental formula. Values of 0.000 are non-zero, however not within three decimal places.

Biased Reservoir Sampling, $\epsilon = 0.05$, $\delta = 0.0001$, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.015	0.001	0.016	0.001	0.015	0.001
Update Time	(s)	0.015	0.001	0.016	0.001	0.015	0.001
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	27.075	0.008	27.147	0.007	27.127	0.008
CLR Heap	(MB)	1.353	0.033	1.353	0.033	1.345	0.024

Biased Reservoir Sampling, $\epsilon = 0.05$, $\delta = 0.0001$, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.015	0.001	0.244	0.004	0.490	0.012
Update Time	(s)	0.015	0.001	0.244	0.004	0.490	0.012
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	27.127	0.008	43.944	0.008	62.137	0.024
CLR Heap	(MB)	1.345	0.024	1.341	0.015	1.343	0.018

Table 6.2: Measurements for computing the bounded differential aggregate variance under the incremental approach with biased reservoir sampling. Values of 0.000 are non-zero, however not within three decimal places.

Biased Reservoir Sampling, $\epsilon = 0.01$, $\delta = 0.0001$, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.016	0.001	0.016	0.001	0.016	0.001
Update Time	(s)	0.016	0.001	0.016	0.001	0.016	0.001
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	28.323	0.013	28.211	0.008	28.556	0.109
CLR Heap	(MB)	2.827	0.040	2.819	0.035	2.855	0.201

Biased Reservoir Sampling, $\epsilon = 0.01$, $\delta = 0.0001$, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.016	0.001	0.256	0.002	0.508	0.018
Update Time	(s)	0.016	0.001	0.256	0.002	0.508	0.018
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	28.556	0.109	45.285	0.011	66.384	0.061
CLR Heap	(MB)	2.855	0.201	2.810	0.016	2.804	0.009

Table 6.3: Measurements for computing the bounded differential aggregate variance under the incremental approach with biased reservoir sampling. Values of 0.000 are non-zero, however not within three decimal places.

Expiration List, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.019	0.002	0.021	0.004	0.021	0.004
Update Time	(s)	0.019	0.002	0.021	0.004	0.021	0.004
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	31.972	1.456	58.359	3.722	115.545	8.522
CLR Heap	(MB)	4.798	1.663	25.255	3.829	82.110	8.529

Expiration List, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.021	0.004	0.363	0.021	0.762	0.028
Update Time	(s)	0.021	0.004	0.363	0.021	0.762	0.028
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	115.545	8.522	133.397	12.242	154.812	13.234
CLR Heap	(MB)	82.110	8.529	81.624	13.437	82.385	13.253

Table 6.4: Measurements for computing the bounded differential aggregate variance under the incremental approach with expiration lists. Values of 0.000 are non-zero, however not within three decimal places.

Panes, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.016	0.000	0.016	0.000	0.016	0.000
Update Time	(s)	0.016	0.000	0.016	0.000	0.016	0.000
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	27.095	0.008	26.998	0.205	27.262	0.008
CLR Heap	(MB)	1.304	0.046	1.338	0.200	1.301	0.041

Panes, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.016	0.000	0.241	0.002	0.476	0.004
Update Time	(s)	0.016	0.000	0.241	0.002	0.476	0.004
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	27.262	0.008	43.764	0.009	62.063	0.011
CLR Heap	(MB)	1.301	0.041	1.287	0.020	1.275	0.004

Table 6.5: Measurements for computing the bounded differential aggregate variance under the incremental approach with panes. Values of 0.000 are non-zero, however not within three decimal places.

Variance Histogram, $\epsilon = 0.05$, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.027	0.001	0.034	0.002	0.037	0.003
Update Time	(s)	0.027	0.001	0.034	0.002	0.037	0.003
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	31.474	1.247	32.700	1.216	33.850	1.666
CLR Heap	(MB)	3.428	1.128	4.446	1.370	5.460	1.297

Variance Histogram, $\epsilon = 0.05$, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.037	0.003	0.529	0.017	1.097	0.025
Update Time	(s)	0.037	0.003	0.529	0.017	1.097	0.025
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	33.850	1.666	76.399	7.443	116.523	10.100
CLR Heap	(MB)	5.460	1.297	23.298	7.254	44.885	10.031

Table 6.6: Measurements for computing the bounded differential aggregate variance under the incremental approach with a variance histogram. Values of 0.000 are non-zero, however not within three decimal places.

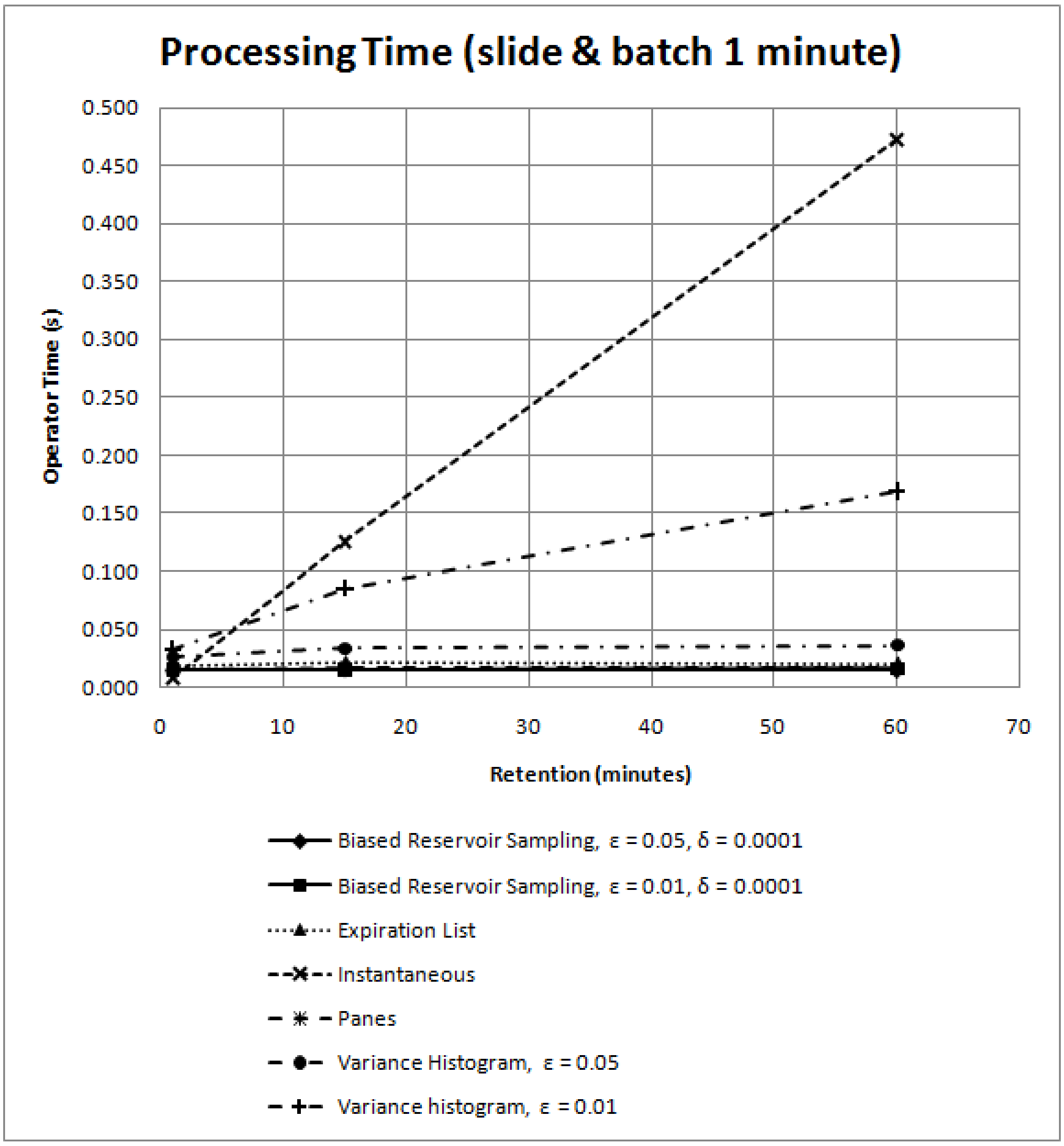


Figure 6.1: Processing time relationship as operator time in seconds and retention in minutes for the bounded differential aggregate variance. The plotted data points are the means of the operator time for fixed slide and variable retention based on the tabulated results.

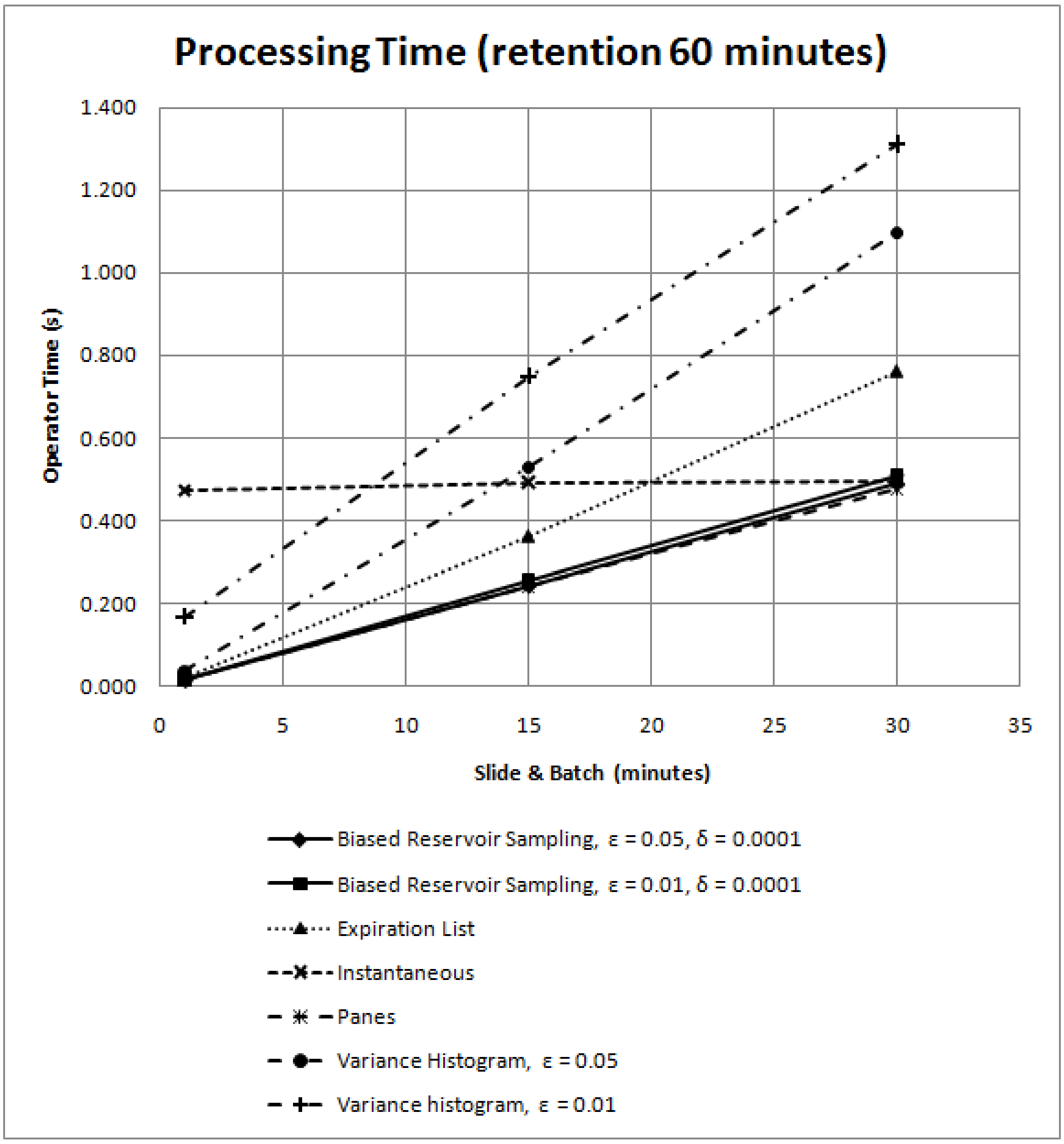


Figure 6.2: Processing time relationship as operator time in seconds and slide in minutes for the bounded differential aggregate variance. The plotted data points are the means of the operator time for variable slide and fixed retention based on the tabulated results.

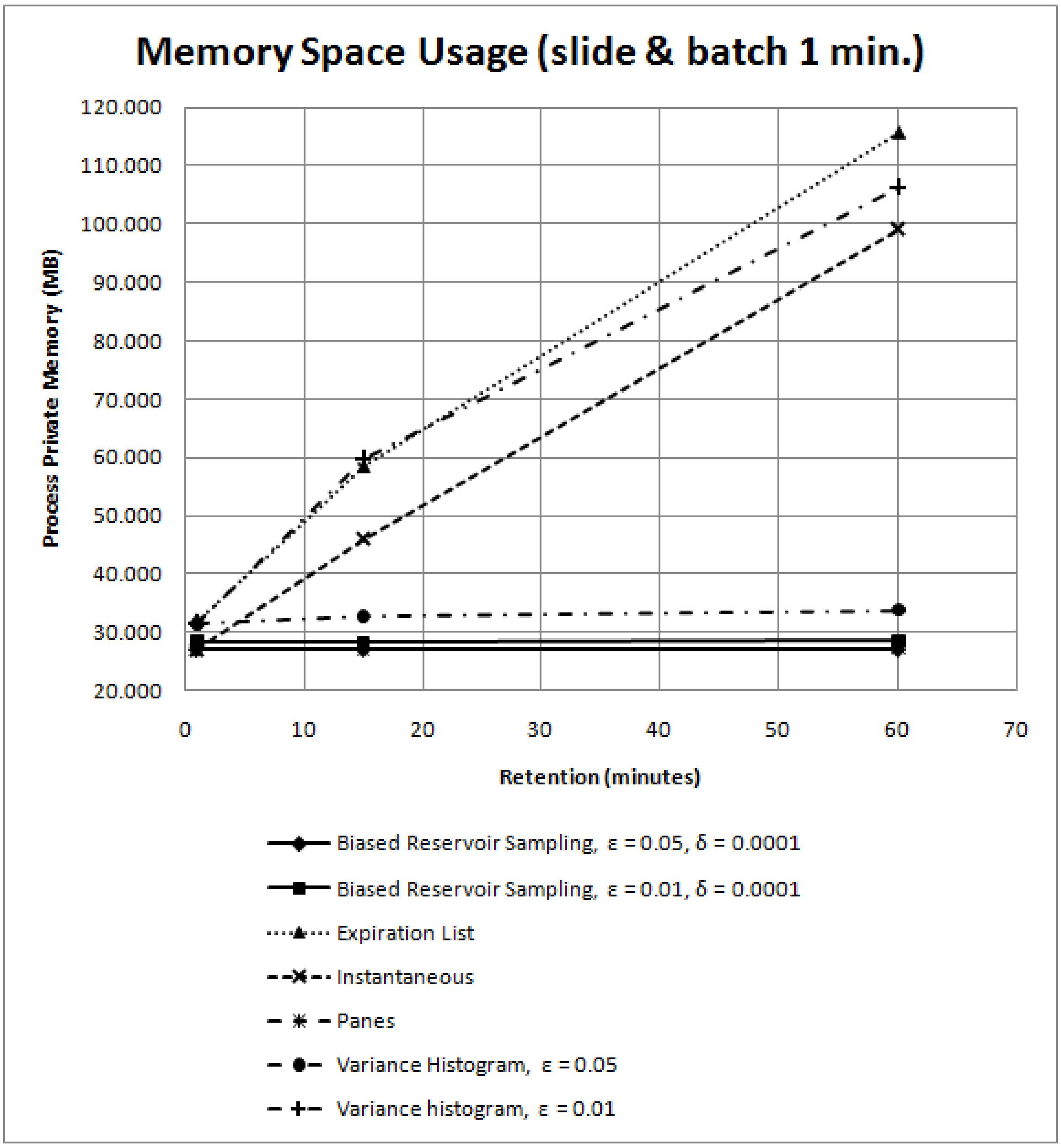


Figure 6.3: Memory space usage relationship as process private memory in megabytes and retention in minutes for the bounded differential aggregate variance. The plotted data points are the means of the process private memory for fixed slide and variable retention based on the tabulated results.

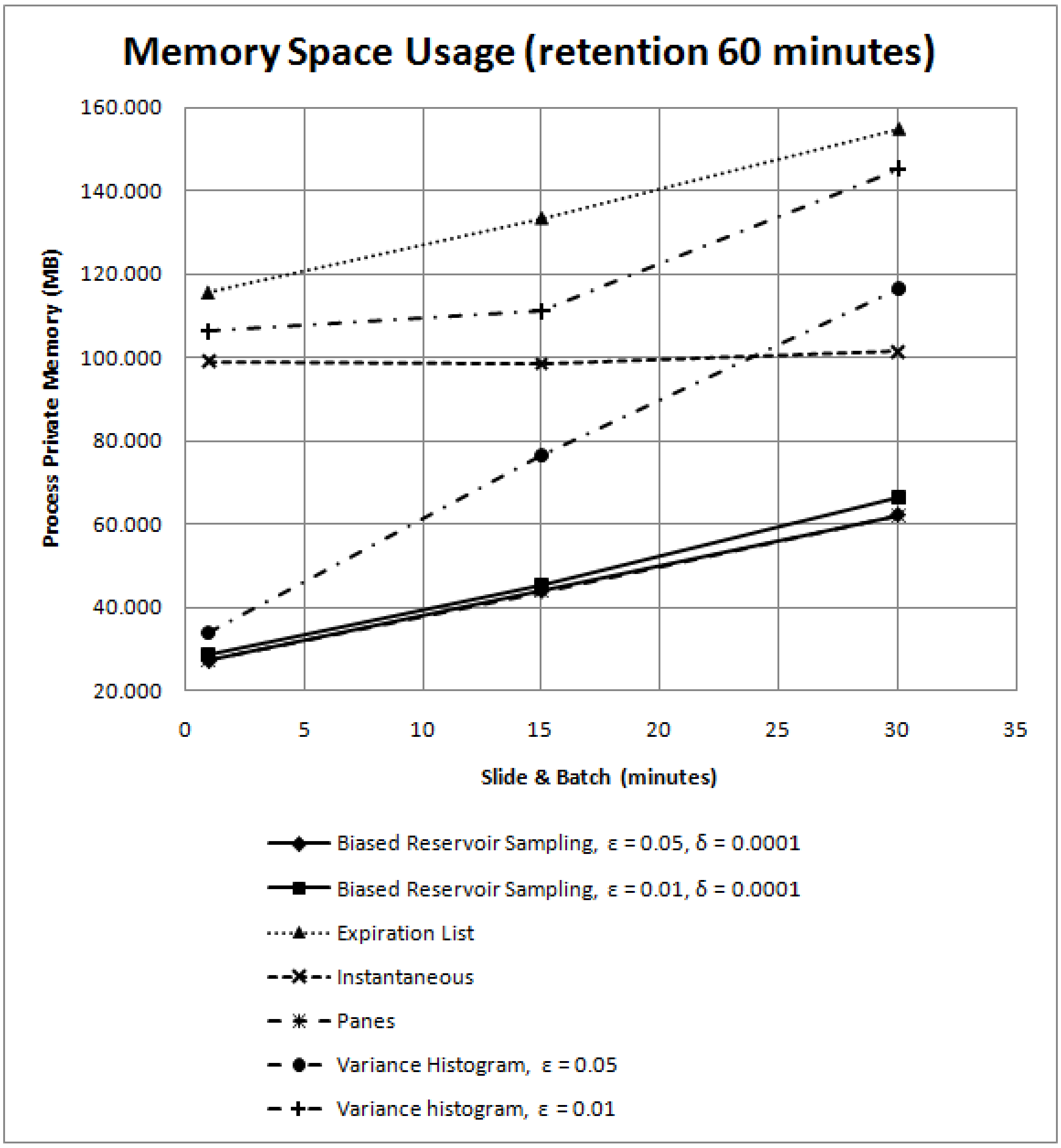


Figure 6.4: Memory space usage relationship as process private memory in megabytes and slide in minutes for the bounded differential aggregate variance. The plotted data points are the means of the process private memory for variable slide and fixed retention based on the tabulated results.

Variance histogram, $\epsilon = 0.01$, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.033	0.002	0.086	0.011	0.169	0.020
Update Time	(s)	0.033	0.002	0.086	0.011	0.169	0.020
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	31.569	1.446	59.717	3.696	106.343	12.212
CLR Heap	(MB)	4.687	1.550	21.780	3.057	70.514	11.663

Variance histogram, $\epsilon = 0.01$, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.169	0.020	0.750	0.034	1.311	0.046
Update Time	(s)	0.169	0.020	0.750	0.034	1.311	0.046
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	106.343	12.212	111.124	20.605	145.273	16.022
CLR Heap	(MB)	70.514	11.663	59.493	20.240	70.766	16.591

Table 6.7: Measurements for computing the bounded differential aggregate variance under the incremental approach with a variance histogram. Values of 0.000 are non-zero, however not within three decimal places.

Biased Reservoir Sampling, $\epsilon = 0.05$, $\delta = 0.0001$				
Batch & Slide	(minutes)	1	1	1
Retention	(minutes)	1	15	60
Measurements	(N)	419	419	419
Average		0.00716	0.00854	0.00847
Maximum		0.02758	0.04371	0.03479
Observed δ		0	0	0

Table 6.8: Accuracy of computing the bounded differential aggregate variance under the incremental approach with biased reservoir sampling compared to the instantaneous approach. The relative error is shown by average and maximum observed relative error ϵ. The observed δ is presented too. Values of 0.00000 for average and maximum relative errors are non-zero, however not within five decimal places.

Biased Reservoir Sampling, $\epsilon = 0.01$, $\delta = 0.0001$				
Batch & Slide	(minutes)	1	1	1
Retention	(minutes)	1	15	60
Measurements	(N)	419	419	419
Average		0.00483	0.00216	0.00154
Maximum		0.01753	0.00925	0.00617
Observed δ		0.10501	0	0

Table 6.9: Accuracy of computing the bounded differential aggregate variance under the incremental approach with biased reservoir sampling compared to the instantaneous approach. The relative error is shown by average and maximum observed relative error ϵ. The observed δ is presented too. Values of 0.00000 for average and maximum relative errors are non-zero, however not within five decimal places.

6.2.2 Accuracy

Regarding accuracy, only biased reservoir sampling and variance histograms will produce approximated aggregates. Their results compared to output of the instantaneous approach are shown below in terms of observed maximum and average relative errors and observed success probability. Observed relative error and observed success probability are discussed in chapter 5. The accuracy results of biased reservoir sampling are in tables 6.8 and 6.9. The accuracy results of variance histograms are in tables 6.10 and 6.11.

Biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ shows that the maximum observed error does not exceed the relative error bound $\epsilon = 0.05$ and thus satisfies the accuracy requirements. The observed error on average is far less than the maximum observed error. Biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$ shows that the observed error can exceed $\epsilon = 0.01$, though on average it is far less. Note that the sample size exceeds lower retention sizes. This causes it to contains data from previous retentions with higher probability and more likely increase the observed error (see chapter 4 in particular section 4.5). Note that the sample size of $O(\epsilon^{-2} * \log(\epsilon\delta)^{-1})$ is mentioned for determining quantiles and, in absence of a better choice, is employed for variance as well.

Variance histograms with $\epsilon = 0.05$ have smaller observed errors compared to biased reservoir sampling. The maximum observed error by far does not exceed $\epsilon = 0.05$. Without a doubt it satisfies the accuracy requirements. The observed error is very small on average. Similar arguments hold for variance with $\epsilon = 0.01$. Interestingly the errors reported at both accuracy levels are similar. It seems that the increased accuracy level has little effect on observed errors, even though at higher accuracy levels the histograms need to maintain many more buckets increasing processing time and memory space usage (see table 6.6 and 6.7 and graphs 6.1, 6.2, 6.3 and 6.4). This is an unexpected phenomenon that could not be explained.

From the accuracy results is seems that good accuracy can be achieved. On average the observed errors are clearly below the relative error bound and within the given success probability. An advantage of variance histograms is their determinism regarding the relative error bound as opposed to sampling, although sampling has better (quantitative) performance as is clear from section 6.2.1. It should be kept in mind that during evaluation uniform random values were aggregated over. Other distributions can yield different results.

Variance Histogram, ϵ = 0.05				
Batch & Slide	(minutes)	1	1	1
Retention	(minutes)	1	15	60
Measurements	(N)	419	419	419
Average		0.00010	0.00181	0.00098
Maximum		0.00050	0.00607	0.00516
Delta		0	0	0

Table 6.10: Accuracy of computing the bounded differential aggregate variance under the incremental approach with variance histograms compared to the instantaneous approach. The relative error is shown by average and maximum observed relative error ϵ. The (observed) δ is 0. Values of 0.00000 for average and maximum relative errors are non-zero, however not within five decimal places.

Variance Histogram, ϵ = 0.01				
Batch & Slide	(minutes)	1	1	1
Retention	(minutes)	1	15	60
Measurements	(N)	419	419	419
Average		0.00010	0.00180	0.00114
Maximum		0.00054	0.00561	0.00416
Delta		0	0	0

Table 6.11: Accuracy of computing the bounded differential aggregate variance under the incremental approach with variance histograms compared to the instantaneous approach. The relative error is shown by average and maximum observed relative error ϵ. The (observed) δ is 0. Values of 0.00000 for average and maximum relative errors are non-zero, however not within five decimal places.

6.3 Results: Quantiles

This section shows the empirical results for the unbounded non-differential aggregate quantiles summarized in graphs and tables. A discussion of the results in text follows. The quantitative performance in terms of actual processing time and memory space usage is treated in section 6.3.1. The qualitative performance in terms of observed accuracy is treated in section 6.3.2.

6.3.1 Performance

The aggregate quantiles is evaluated under the incremental approach; see results in table 6.12. Furthermore it is evaluated under the incremental approach by means of biased reservoir sampling (see section 4.5), expiration lists (see section 4.3) and Arasu-Manku fixed sketches (see section 4.7). See tables 6.13 and 6.14 for results on biased reservoir sampling; see table 6.15 for results on expiration lists; see tables 6.16 and 6.17 for results on Arasu-Manku fixed sketches. The relationships of processing time versus retention and slide are shown in the graphs 6.5 and 6.6. Similarly relationships of memory space usage versus retention and slide are shown in the graphs 6.7 and 6.8.

The instantaneous approach (see table 6.12 and graphs 6.5, 6.6, 6.7 and 6.8) shows processing time increases more as the retention increases and memory space usage grows linearly. This confirms expectations as quantiles require: sorting taking time at most $O(R * \log R)$ and a pass over the sorted data to obtain the quantiles (as described in chapter 3). Processing time increases more (super-linear) as retention increases. Keeping a large window capturing the retention and a sorted (multi)set over the same number of elements, explains the linear growth of memory space usage. An increase in slide has hardly any effect on processing time and memory space usage. This confirms the theoretical model of chapter 3 where the processing time $T(R)$ and memory space usage $S(R)$ of the instantaneous approach are independent of the slide. The CLR heap uses quite some memory space, which is explained by keeping the sorted (multi)set of elements.

The incremental approach using biased reservoir sampling with $\delta = 0.01$ for $\epsilon = 0.05$ and $\epsilon = 0.01$ (see table 6.13 and 6.14 and graphs 6.5, 6.6, 6.7 and 6.8) shows constant processing time and memory space usage as the retention increases and both growing linearly as the slide and batch size increase. The sample size, $O(\frac{1}{\epsilon^2} * \log \frac{1}{\epsilon * \delta})$, is independent of the retention, explaining the constant memory space usage as the retention increases. As the slide and batch size increase, more elements are processed and kept in memory. This explains the (linear) growth. For sampling with $\epsilon = 0.05$ the results show clear processing time $T(R, S, B)$ benefits up to a slide $S \approx \frac{1}{2} * R$. For sampling with $\epsilon = 0.01$ the results show processing time benefits by $T(R, S, B)$ are lost at a slide $S \approx \frac{2}{5} * R$. Sampling shows a clear memory space usage benefit by $S(B, R)$ up to a slide $S \approx \frac{1}{2} * R$. The standard deviation is low, indicating a rather stable usage of system resources.

The incremental approach using expiration lists (see table 6.15 and graphs 6.5, 6.6, 6.7 and 6.8) shows logarithmically increasing processing time and anomalous memory space usage behavior as the retention increases. The processing time can be explained by inserting new data and removing expired data in $O(\log R)$ as described in section 4.3 from a (balanced) binary search tree to keep the retention value-ordered. The memory space usage could not be explained and might be an issue with the third-party implementation of the binary search tree. As the slide and batch size increase, so does the number of elements processed and kept in memory. This is obvious from the linear growth of processing time. Again memory space usage shows anomalous behavior. Most likely the increased batch size is overshadowed by the memory space usage of the time-ordered linked list and value-ordered (balanced) binary search tree consuming space $O(2 * R)$. At about a slide $S \approx \frac{1}{5} * R$ the overhead of maintaining the time-ordered and value-ordered data is $S * T_u(R)$ and exceeds the processing time of the instantaneous approach $T(R)$. Clearly, no memory space usage $S(B, R)$ benefit is expected. The very high variance in memory space usage is worrisome, suggesting a rather unstable use of system resources.

The incremental approach using the Arasu-Manku fixed sketches with $\epsilon = 0.05$ and $\epsilon = 0.01$ (see table 6.16 and 6.17 and graphs 6.5, 6.6, 6.7 and 6.8) shows near constant processing time and memory space usage similar to sampling with $\epsilon = 0.05$. The poly-logarithmic space complexity of the Arasu-Manku fixed sketches requires little memory space usage and apparently takes little processing time. As the slide and batch size increase, the processing time grows linearly (exceeding sampling) and memory space usage grows linearly (close to sampling with $\epsilon = 0.05$). For Arasu-Manku fixed sketches with $\epsilon = 0.05$ the results show processing time $T(R, S, B)$ benefits are lost at a slide $S \approx \frac{1}{3} * R$. For Arasu-Manku fixed sketches with $\epsilon = 0.01$ the results show processing time $T(R, S, B)$ benefits are lost at a slide $S \approx \frac{1}{4} * R$. The Arasu-Manku fixed sketches show clear memory space usage $S(B, R)$ benefits up to a slide $S \approx \frac{1}{2} * R$. Note that the fixed sketches were used during evaluation, though for true support of

Instantaneous, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.018	0.001	0.437	0.013	2.794	0.064
Process Private Memory	(MB)	29.055	0.105	60.167	3.179	156.028	14.936
CLR Heap	(MB)	1.612	0.267	9.191	2.663	42.922	13.624

Instantaneous, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	2.794	0.064	2.783	0.083	2.801	0.134
Process Private Memory	(MB)	156.028	14.936	157.910	14.578	151.636	15.184
CLR Heap	(MB)	42.922	13.624	43.970	13.217	44.301	13.675

Table 6.12: Measurements for computing the unbounded non-differential aggregate quantiles under the instantaneous approach. Values of 0.000 are non-zero, however not within three decimal places.

variable sized retention (e.g. time-based; see section 2.3) an adapted version of Arasu-Manku sketches is required that requires more memory space and processing time in practice (see section 4.7).

In general it can be observed that the instantaneous approach shows a slight super-linear growth of processing time and linear growth of memory space usage as the retention increases, independent of the slide (as expected). The incremental approach necessarily has performance grow as the slide and batch size increase, because larger batches with more elements are processed. Sampling performs best, though its approximation is randomized / probablistic. Arasu-Manku fixed sketches provide deterministic approximations, at the expense of processing time with respect to sampling. The expiration list only has processing time benefits as the slide is sufficiently small. In general processing time of the incremental approach has lost its benefit at about a slide of half the retention. Query time $T_q(R)$ is clearly larger for quantiles than it is for variance, as expected by non-differentiality of quantiles versus differentiality of variance. The update time $S * T_u(R)$ for quantiles far exceeds the query time (except for expiration lists) and exceeds the processing time of the instantaneous approach as the slide and batch size are sufficiently large. Memory space usage benefits are possible beyond a slide of half the retention when approximations are used, e.g. sampling or Arasu-Manku fixed sketches. It should be kept in mind that during evaluation uniform random values were aggregated over. Other distributions can yield different results, especially if they are favored or disfavored by certain algorithms.

Biased Reservoir Sampling, $\epsilon = 0.05$, $\delta = 0.0001$, **fixed slide**							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.053	0.001	0.052	0.001	0.053	0.001
Update Time	(s)	0.052	0.001	0.052	0.001	0.052	0.001
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	33.479	0.762	28.397	0.671	28.135	0.679
CLR Heap	(MB)	2.140	0.403	2.318	0.370	2.189	0.373

Biased Reservoir Sampling, $\epsilon = 0.05$, $\delta = 0.0001$, **fixed retention**							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.053	0.001	0.742	0.012	1.535	0.044
Update Time	(s)	0.052	0.001	0.741	0.012	1.534	0.044
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	28.135	0.679	46.502	0.388	64.593	0.204
CLR Heap	(MB)	2.189	0.373	2.484	0.568	2.068	0.226

Table 6.13: Measurements for computing the unbounded non-differential aggregate quantiles under the incremental approach with biased reservoir sampling. Values of 0.000 are non-zero, however not within three decimal places.

Biased Reservoir Sampling, $\epsilon = 0.01$, $\delta = 0.0001$							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.145	0.014	0.151	0.013	0.142	0.013
Update Time	(s)	0.122	0.014	0.128	0.013	0.119	0.014
Query Time	(s)	0.023	0.001	0.023	0.001	0.022	0.001
Process Private Memory	(MB)	73.658	5.955	73.921	3.161	73.314	4.043
CLR Heap	(MB)	39.130	5.272	34.437	2.158	34.392	3.030

Biased Reservoir Sampling, $\epsilon = 0.01$, $\delta = 0.0001$, **fixed retention**							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.142	0.013	1.731	0.072	3.425	0.074
Update Time	(s)	0.119	0.014	1.709	0.072	3.402	0.074
Query Time	(s)	0.022	0.001	0.023	0.001	0.023	0.001
Process Private Memory	(MB)	73.314	4.043	71.983	5.936	90.755	4.895
CLR Heap	(MB)	34.392	3.030	21.000	5.857	22.231	4.907

Table 6.14: Measurements for computing the unbounded non-differential aggregate quantiles under the incremental approach with biased reservoir sampling. Values of 0.000 are non-zero, however not within three decimal places.

Expiration List, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.052	0.002	0.207	0.029	0.373	0.036
Update Time	(s)	0.050	0.002	0.166	0.029	0.200	0.029
Query Time	(s)	0.002	0.000	0.042	0.003	0.173	0.011
Process Private Memory	(MB)	38.158	1.725	153.448	9.684	274.855	103.508
CLR Heap	(MB)	5.162	2.564	118.289	9.743	228.682	101.676

Expiration List, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.373	0.036	3.207	0.290	6.301	0.564
Update Time	(s)	0.200	0.029	3.035	0.289	6.122	0.559
Query Time	(s)	0.173	0.011	0.172	0.006	0.179	0.024
Process Private Memory	(MB)	274.855	103.508	242.944	58.358	256.958	71.698
CLR Heap	(MB)	228.682	101.676	178.021	58.156	172.679	71.525

Table 6.15: Measurements for computing the unbounded non-differential aggregate quantiles under the incremental approach with expiration lists. Values of 0.000 are non-zero, however not within three decimal places.

Arasu Manku Fixed Sketch, $\epsilon = 0.05$, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.148	0.002	0.141	0.003	0.139	0.002
Update Time	(s)	0.148	0.002	0.141	0.003	0.139	0.002
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	29.976	0.718	28.056	0.564	30.492	0.274
CLR Heap	(MB)	2.377	0.528	1.908	0.223	1.635	0.087

Arasu Manku Fixed Sketch, $\epsilon = 0.05$, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.139	0.002	2.191	0.072	4.099	0.152
Update Time	(s)	0.139	0.002	2.191	0.072	4.099	0.152
Query Time	(s)	0.000	0.000	0.000	0.000	0.000	0.000
Process Private Memory	(MB)	30.492	0.274	47.469	0.417	63.154	0.461
CLR Heap	(MB)	1.635	0.087	1.651	0.091	1.630	0.085

Table 6.16: Measurements for computing the unbounded non-differential aggregate quantiles under the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. Values of 0.000 are non-zero, however not within three decimal places.

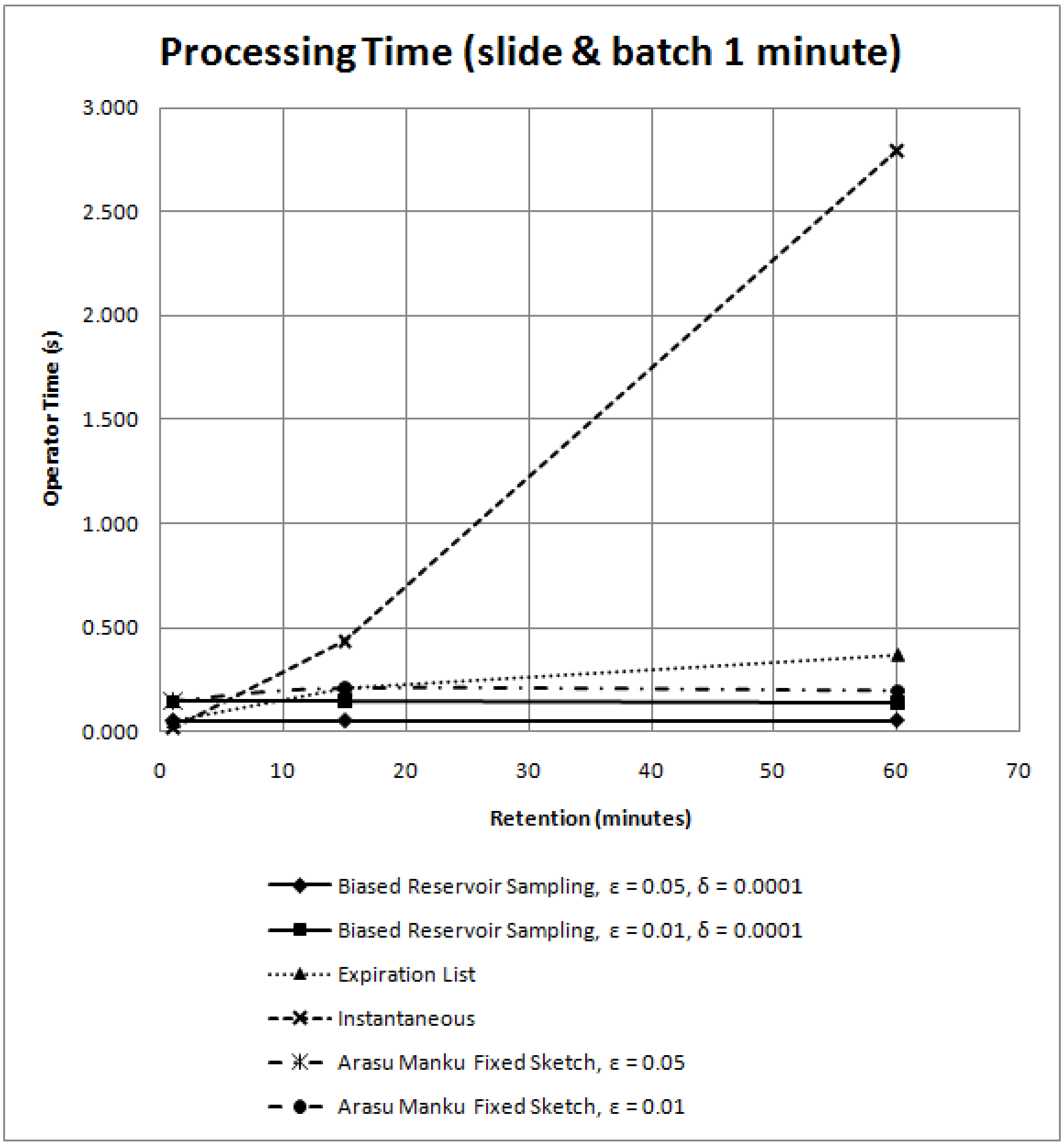


Figure 6.5: Processing time relationship as operator time in seconds and retention in minutes for the unbounded non-differential aggregate quantiles. The plotted data points are the means of the operator time for fixed slide and variable retention based on the tabulated results.

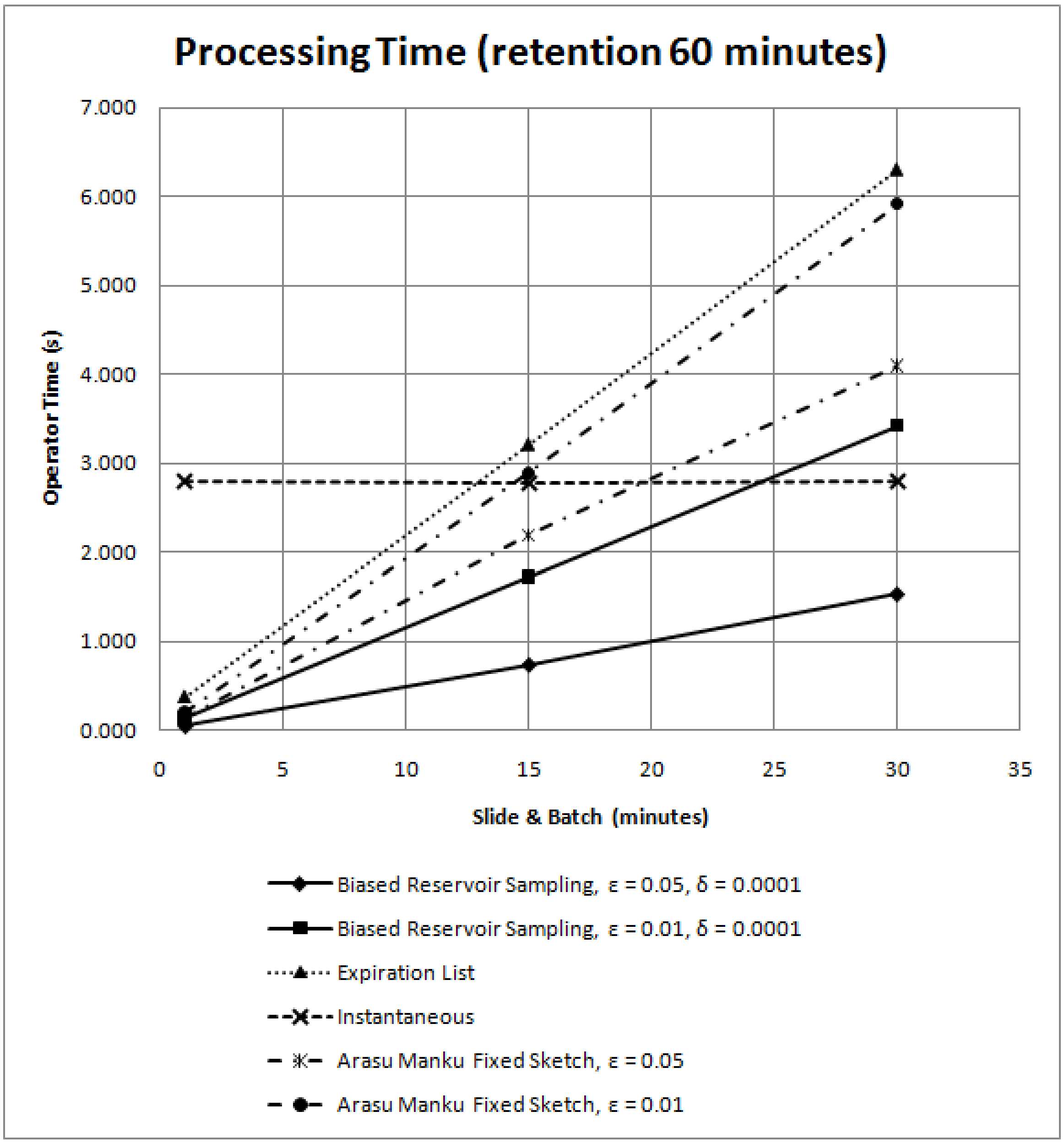


Figure 6.6: Processing time relationship as operator time in seconds and slide in minutes for the unbounded non-differential aggregate quantiles. The plotted data points are the means of the operator time for variable slide and fixed retention based on the tabulated results.

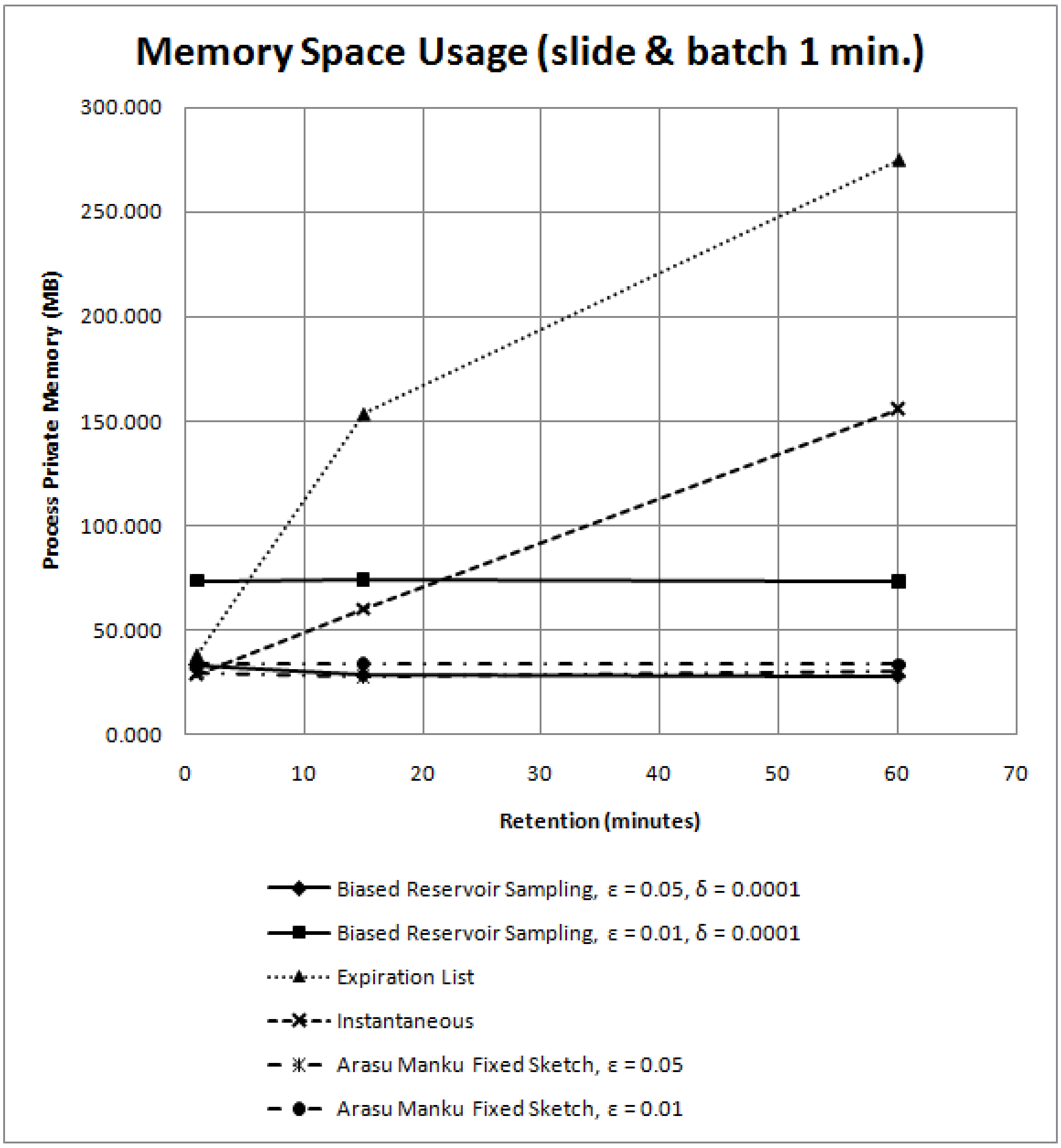


Figure 6.7: Memory space usage relationship as process private memory in megabytes and retention in minutes for the unbounded non-differential aggregate quantiles. The plotted data points are the means of the process private memory for fixed slide and variable retention based on the tabulated results.

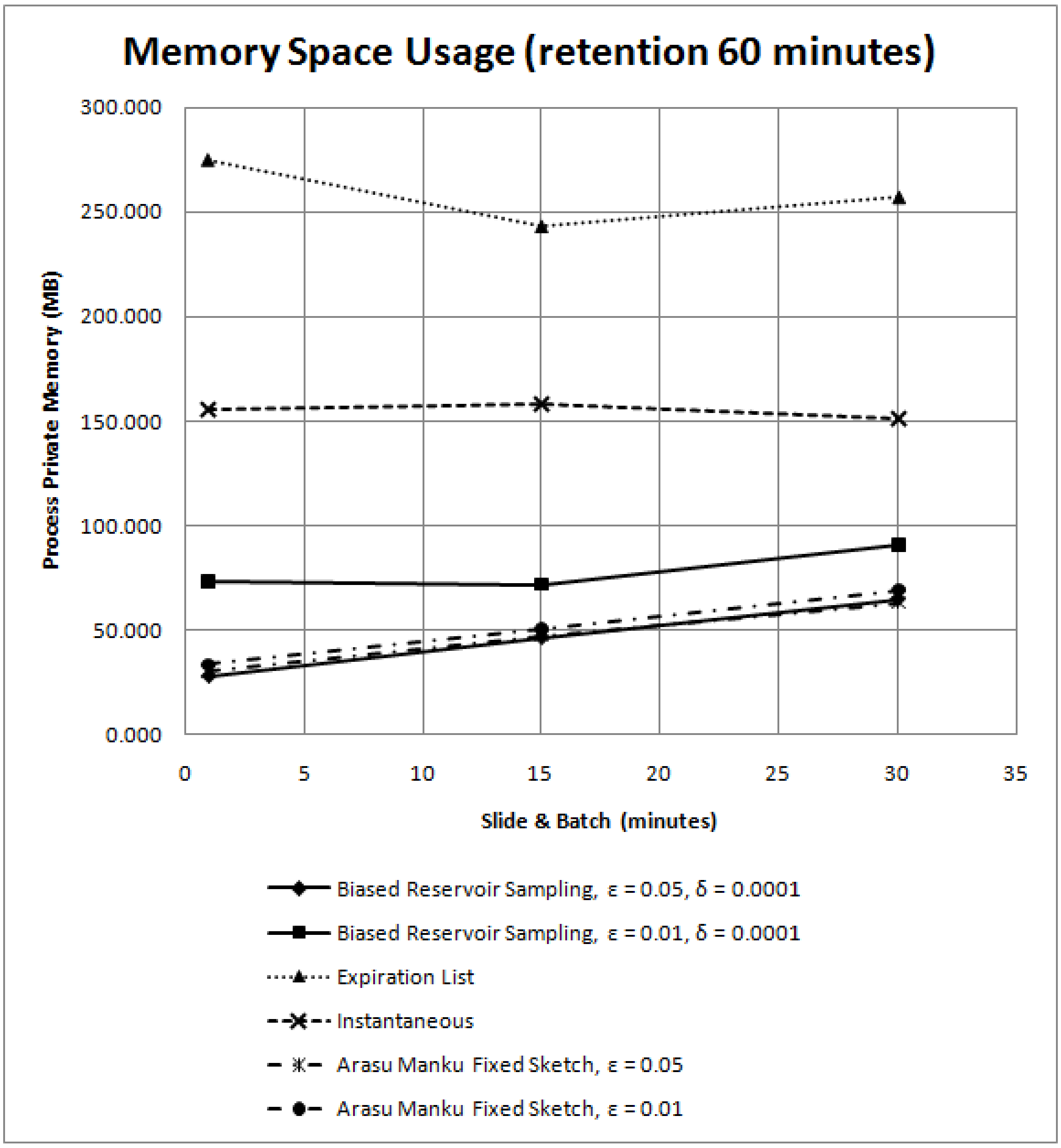


Figure 6.8: Memory space usage relationship as process private memory in megabytes and slide in minutes for the unbounded non-differential aggregate quantiles. The plotted data points are the means of the process private memory for variable slide and fixed retention based on the tabulated results.

Arasu Manku Fixed Sketch, $\epsilon = 0.01$, fixed slide							
Batch & Slide	(minutes)	1		1		1	
Retention	(minutes)	1		15		60	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.144	0.004	0.209	0.007	0.196	0.005
Update Time	(s)	0.142	0.004	0.207	0.007	0.195	0.005
Query Time	(s)	0.002	0.000	0.002	0.000	0.002	0.001
Process Private Memory	(MB)	33.539	1.502	33.965	1.316	33.607	0.992
CLR Heap	(MB)	4.684	1.652	4.646	1.102	4.542	0.866

Arasu Manku Fixed Sketch, $\epsilon = 0.01$, fixed retention							
Retention	(minutes)	60		60		60	
Batch & Slide	(minutes)	1		15		30	
Measurements	(N)	50		50		50	
		Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Operator Time	(s)	0.196	0.005	2.892	0.022	5.918	0.424
Update Time	(s)	0.195	0.005	2.890	0.022	5.917	0.424
Query Time	(s)	0.002	0.001	0.002	0.000	0.002	0.001
Process Private Memory	(MB)	33.607	0.992	50.446	1.065	69.279	1.424
CLR Heap	(MB)	4.542	0.866	4.052	1.029	4.068	1.232

Table 6.17: Measurements for computing the unbounded non-differential aggregate quantiles under the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. Values of 0.000 are non-zero, however not within three decimal places.

Biased Reservoir Sampling, $\epsilon = 0.05$, $\delta = 0.0001$						
Batch & Slide	(minutes)			1		
Retention	(minutes)			1		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00009	0.00347	0.00415	0.00360	0.00010
Maximum		0.00068	0.01284	0.01596	0.01503	0.00094
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			15		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00014	0.00412	0.00475	0.00404	0.00014
Maximum		0.00088	0.01630	0.01798	0.01721	0.00099
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			60		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00016	0.00414	0.00454	0.00408	0.00014
Maximum		0.00119	0.01566	0.01995	0.01854	0.00127
Observed δ		0	0	0	0	0

Table 6.18: Accuracy of computing the unbounded non-differential aggregate quantiles under the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ compared to the instantaneous approach. The observed relative errors at $\phi \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ are shown by average and maximum observed relative error ϵ. The observed δ is presented too. Values of 0.00000 of average and maximum relative errors are non-zero, however not within five decimal places.

6.3.2 Accuracy

Regarding accuracy, biased reservoir sampling and Arasu-Manku fixed sketches will produce approximated aggregates. Their results compared to output of the instantaneous approach are shown below in terms of observed maximum and average relative errors and observed success probability. Observed relative error and observed success probability are discussed in chapter 5. The quantiles were chosen to cover the extremes (minimum and maximum) and the center (toward and including the median) to be able to observe bias of the approximation, hence $\phi \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. The accuracy results of biased reservoir sampling are in tables 6.18 and 6.19. The accuracy results of Arasu-Manku fixed sketches are in tables 6.20 and 6.21.

Biased reservoir sampling with $\epsilon = 0.05$ and $\epsilon = 0.01$ and $\delta = 0.0001$ satisfied the relative error bound of the accuracy requirements with the given success probability. Interestingly the observed errors for the extreme values of ϕ, the minimum and maximum, have a clearly lower maximum and average than values of ϕ approaching the median. The accuracy results for a sample size of $O(\frac{1}{\epsilon^2} * \log \frac{1}{\epsilon * \delta})$ confirm results from literature [5].

Arasu-Manku fixed sketches with $\epsilon = 0.05$ and $\epsilon = 0.01$ satisfied the relative error bound of the accuracy requirements as expected. In general the average and maximum observed errors are clearly larger than biased reservoir sampling, though they are obtained deterministically rather than randomized / probabilistically. The errors for the various values of ϕ are similar, except for ϕ approaching the maximum where the error is far lower. This might be explained by the fact that block level sketches of Arasu-Manku fixed sketches explicitly contain the maximum over the block though not the minimum (see chapter 4 in particular section 4.7). Hence the overall maximum is approximated better than the overall minimum. Nevertheless the approximation is still robust, which is favorable as quantiles belong to the exemplary aggregates that require robust approximation (as argued for in chapter 3).

From the accuracy results is seems that good accuracy can be achieved. On average the observed errors are clearly below the relative error bound and within the given success probability. An advantage over Arasu-Manku fixed sketches is their determinism regarding the relative error bound as opposed to sampling. It should be kept in mind that during evaluation uniform random values were aggregated over. Other distributions can yield different results.

Biased Reservoir Sampling, $\epsilon = 0.01$, $\delta = 0.0001$						
Batch & Slide	(minutes)			1		
Retention	(minutes)			1		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00000	0.00229	0.00260	0.00241	0.00000
Maximum		0.00005	0.00931	0.00941	0.00894	0.00005
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			15		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00000	0.00066	0.00065	0.00058	0.00000
Maximum		0.00002	0.00262	0.00249	0.00228	0.00002
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			60		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00000	0.00062	0.00069	0.00065	0.00000
Maximum		0.00002	0.00202	0.00221	0.00186	0.00002
Observed δ		0	0	0	0	0

Table 6.19: Accuracy of computing the unbounded non-differential aggregate quantiles under the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ compared to the instantaneous approach. The observed relative errors at $\phi \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ are shown by average and maximum observed relative error ϵ. The observed δ is presented too. Values of 0.00000 of average and maximum relative errors are non-zero, however not within five decimal places.

Arasu Manku Fixed Sketch, $\epsilon = 0.05$						
Batch & Slide	(minutes)			1		
Retention	(minutes)			1		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00226	0.00252	0.00122	0.00249	0.00019
Maximum		0.00504	0.00967	0.01061	0.01258	0.00151
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			15		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00254	0.00247	0.00268	0.00313	0.00003
Maximum		0.00475	0.00845	0.01201	0.00863	0.00087
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			60		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00291	0.00275	0.00224	0.00227	0.00004
Maximum		0.00509	0.00775	0.00770	0.00702	0.00166
Observed δ		0	0	0	0	0

Table 6.20: Accuracy of computing the unbounded non-differential aggregate quantiles under the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$ compared to the instantaneous approach. The observed relative errors at $\phi \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ are shown by average and maximum observed relative error ϵ. The observed δ is presented too. Values of 0.00000 of average and maximum relative errors are non-zero, however not within five decimal places.

Arasu Manku Fixed Sketch, $\epsilon = 0.01$						
Batch & Slide	(minutes)			1		
Retention	(minutes)			1		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00036	0.00020	0.00023	0.00196	0.00004
Maximum		0.00140	0.00166	0.00129	0.00868	0.00052
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			15		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00034	0.00059	0.00069	0.00066	0.00002
Maximum		0.00079	0.00260	0.00349	0.00288	0.00020
Observed δ		0	0	0	0	0
Batch & Slide	(minutes)			1		
Retention	(minutes)			60		
Measurements	(N per ϕ)			419		
Quantile	ϕ	0.0	0.25	0.5	0.75	1.0
Average		0.00035	0.00054	0.00045	0.00046	0.00000
Maximum		0.00076	0.00216	0.00175	0.00177	0.00007
Observed δ		0	0	0	0	0

Table 6.21: Accuracy of computing the unbounded non-differential aggregate quantiles under the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$ compared to the instantaneous approach. The observed relative errors at $\phi \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ are shown by average and maximum observed relative error ϵ. The observed δ is presented too. Values of 0.00000 of average and maximum relative errors are non-zero, however not within five decimal places.

6.4 Wrap-up

As a general remark on the results, it should be noted that batch size is equal to the slide during the measurements and hence the results. In the preliminary section of this chapter, it was concluded that it most likely affects memory space usage $S(B, R)$ which directly depends on the batch size and synopsis structure covering the retention. From the theoretical model, the batch size seems insignificant regarding processing time $T(R, S, B) = S * T_u(R) + T_q(R)$ (see chapter 3). The performance of larger slides and smaller batches cannot be observed.

For variance using sampling or panes in the incremental approach has the best (and similar) performance, especially low memory space usage even at larger slides. An advantage of panes is their exact computation. When the number of panes is sufficiently small, similar results can be obtained for other bounded aggregates. Variance histograms approximate variance, yielding memory space benefits at lower levels of accuracy or for very large retention sizes at higher levels of accuracy and small slide / batch sizes. Both sampling and variance histograms have good accuracy, though sampling achieves it probablistically. The expiration list only has processing time benefits for small slides / batch sizes. Processing time benefits of the incremental approach are lost at about a retention of half the slide, whereas clear memory space usage benefits can still be obtained at that point.

For quantiles using sampling or Arasu-Manku fixed sketches in the incremental approach has the best performance, especially low memory space usage even at larger slides. Sampling and Arasu-Manku fixed sketches have good accuracy, though sampling achieves it probablistically. Sampling has a bias to lower errors for the extreme values (minimum and maximum) of ϕ and Arasu-Manku fixed sketches have a bias to lower errors for the high values of ϕ (the maximum). The expiration list has processing time benefits for small slides. Processing time benefits of the incremental approach are lost at about a retention of half the slide, whereas clear memory space usage benefits can still be obtained at that point.

Most importantly both the instantaneous and incremental approach have processing time and / or memory space usage benefits depending on the retention, size and batch size. Note that the results are tied to particular implementations and the environment. Moreover the distribution of the data over which is aggregates is fixed to uniform random values. In the next chapter 7 the empirical results of the environment are generalized by regression models to serve as empirical approximations in the theoretical model of chapter 3.

The following **guidelines** regarding the performance in practice (from empirical results) are derived from this chapter:

- The trivial synopsis structure, expiration list, provides processing time benefits for bounded differential aggregates, if the slide is sufficiently small.
- Panes provide the best processing time and memory space usage for exact computation of bounded (in particular differential) aggregates, if the number of panes is kept small.
- Biased reservoir sampling with a sample size of $O(\frac{1}{\epsilon^2} * \log \frac{1}{\epsilon * \delta})$ achieves good accuracy (probabilistically) at low processing time and memory space usage.
- Variance histograms achieve good accuracy (deterministically) and provide memory space usage benefits as the slide is sufficiently small and the retention is sufficiently large (bias toward very large retention sizes).
- Arasu-Manku fixed sketches achieve good accuracy (deterministically) and provide memory space usage benefits similar to biased reservoir sampling.
- As the slide (and batch size) gets sufficiently large with respect to the retention, the instantaneous approach outperforms the incremental approach regarding processing time and / or memory space usage.

Chapter 7

Empirical Regression Models

This chapter describes the analysis performed with SPSS 17 on the empirical results of processing time and memory space usage results (see chapter 6) to obtain regression models, generalizing the results on the environment. The regression models provide approximations of performance (dependent variable) for an aggregate and approach. They are parameterized on the size of the retention and slide (independent variables). The regression models effectively are empirical approximations for $T(R)$ and $S(R)$ in the instantaneous approach and for $T(R, S, B)$ and $S(B, R)$ in the incremental approach discussed in the theoretical model of chapter 3.

The chapter starts in section 7.1 with a discussion of the regression analysis used to derive regression models from the empirical results. It discusses the analysis, its assumptions and how the models are validated independently. Next the regression models for processing time and memory space usage are presented. The models for the bounded differential aggregate variance are discussed in section 7.2. Next the models for the unbounded non-differential aggregate quantiles are discussed in section 7.3. The chapter ends with a wrap-up in section 7.5.

7.1 Multiple Linear Regression

7.1.1 Regression Model and Assumptions

A widely used approach to model linear relationships from data is known as multiple linear regression (MLR) [24, 42, 51, 55]. In MLR p predictor or independent variables $X_1, X_2, ..., X_p$ are used to predict the value of a response or dependent variable Y. In a dataset of N observations, the observed values y_i of the dependent variable for the values $x_{i,1}, x_{i,2}, ..., x_{i,p}$ of the independent variables with coefficients $\beta_0, \beta_1, ..., \beta_p$ and error term ϵ_i are modeled as follows:

$$y_i = \beta_0 + \beta_1 * x_{i,1} + \beta_2 * x_{i,2} + ... + \beta_{p-1} * x_{i,p} + \epsilon_i = \sum_{k=0}^{p} \beta_k * x_{i,k} + \epsilon_i \text{ with } x_{i,0} = 1$$

The aim is to estimate the coefficients β_i by $\hat{\beta}_i$ and to estimate y_i by $\hat{y}_i$ in the regression model:

$$\hat{y}_i = \sum_{k=0}^{p} \hat{\beta}_k * x_{i,k} \text{ with } x_{i,0} = 1$$

Let the residual e_i be the difference between the observed value y_i and predicted value $\hat{y}_i$. The regression procedure should minimize the sum of squared residuals (SSE):

$$SSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} e_i^2$$

Note that minimizing SSE is a popular method referred to as ordinary least squares (OLS), though other methods can be used too (e.g. least absolute deviation). The ordinary least squares method has some nice properties [51], like computational efficiency and the $\hat{\beta}_i$ being a best linear unbiased estimator (BLUE) of β_i. This means the expected value of $\hat{\beta}_i$ equals β_i and $\hat{\beta}_i$ has the minimum variance among the unbiased estimators for β_i. As the computations involved in multiple linear regression are tedious to do by hand, they are performed using the statistical software package SPSS version 17. Regression is performed using the enter method, i.e. each independent variable selected is incorporated into the regression model (no step-wise regression). Several assumptions underly the OLS regression model [24, 51, 55, 61]:

1. Accuracy of the data regarding: validity and reliability of the observations, absence of outliers and missing values.
2. Some data for every independent variable.
3. Absence of multicollinearity; no highly correlated independent variables.
4. Linearity in the independent variables and the dependent variable.
5. Homoscedasticity; the variance of the residuals being constant.
6. Independence of residuals of different observations.
7. Normality of the residuals.

Regarding the first assumption, the results from the evaluations are expected to be valid and reliable. The independent variables retention and slide are controlled during the research an pose no problems. The stopwatch functionality providing processing time and performance counters of the operating system providing information on memory space usage are assumed to provide accurate and valid values only. Missing values will not occur. Extreme values require some attention, as the regression model is affected by them. They can show up for a variety of reasons unknown from the results themselves. Their removal is non-obvious and avoided as much as possible; the issue is ignored.

The second assumption regards the amount of results from the evaluation used to construct a regression model. As discussed in chapters 5 on setup and 6 on empirical results, each workload has 50 measurements available for analysis. Per aggregate and per approach 5 workloads (see table 5.4) had performance measured. This should provide enough data. However the variation of retention and slide is limited (see section 5.4) in the interest of time. This is a weakness of the setup that can be mitigated by increasing the variation.

The third assumption requires that the independent variables, retention and slide, are not highly correlated. Retention and slide in the workloads are chosen to reflect certain output semantics. It can safely be assumed there is no apparent correlation. The fourth assumption regards the linearity of the independent variables and dependent variables. The ordinary least squares regression used in this chapter expects such a linear relationship. From the empirical results in chapter 6 both linear and non-linear relationships are observed. The nonlinear relationships (e.g. observed for variance histograms with a low error / high accuracy) probably are not captured well by the regression models. The models likely have (seriously) increased error when extrapolating beyond the data used to construct the models. This is a weakness of the analysis used in this chapter that can be mitigated by a more advanced regression analysis, like non-linear regression [41]. Another possibility is apply transformations to the variables to obtain a linear form relationship, e.g taking the logarithm, square root or reciprocal [55]. If the dependent variable was transformed, predictions from the constructed regression model should have the inverse transformation applied. If an independent variable was transformed, input to the constructed regression model should have the same transformations applied.

The fifth, sixth and seventh assumption relate to the residuals. They need to be assessed for every regression model, while the previous assumptions are clear from the results (e.g. graphs showing relationships). The impact of the last three assumptions is best understood after introducing some terminology used literature [51] on regression analysis. The deviations of the observed values y_i from their mean $\bar{y}$ is captured by the total sum of squares (SST):

$$SST = \sum_{i=1}^{N} (y_i - \bar{y})^2$$

The proportion of variance in the dependent variable explained by the independent variables is captured by the coefficient of determination R^2:

$$R^2 = \frac{SST - SSE}{SST}$$

A good regression model has this value close (or ideally equal) to 1.0. The regression sum of squares (SSR) states the deviations from the estimated (predicted) values $\hat{y}_i$ from the mean $\bar{y}$ of the observed values y_i:

$$SSR = \sum_{i=1}^{N} (\hat{y}_i - \bar{y})^2$$

The mean square error (MSE) and root mean squared error (RMSE) also known as standard error of the estimate:

$$MSE = \frac{SSE}{n-p-1} \text{ and } RMSE = \sqrt{MSE}$$

To assess the statistical significance of the model an F-test is performed with:

$$F^* = \frac{SSR}{MSE} \sim F_{p,n-p-1}$$

Clearly both the coefficient of determination and the statistical tests are affected by the (mean) sum of squared residuals SSE. Underestimating SSE overestimates R^2 and F^*, causing an increased chance of a Type I error of wrongly accepting the model. Overestimating SSE understimates R^2 and F^*, causing an increased chance of a Type II error of wrongly rejecting the model. The estimated standard error for coefficient $\hat{\beta}_i$, denoted by $s_{\hat{\beta}_i}$, depends on (R)MSE and is used in the studentized statistic t^* to test the hypothesis $H_0 : \beta_i = 0$ ($H_a : \beta_i \neq 0$) with confidence level $(1 - \alpha)$:

$$t^* = \frac{\hat{\beta}_i}{s_{\hat{\beta}_i}} \sim t_{n-p-1,\frac{\alpha}{2}}$$

Overestimating t^* (e.g. by underestimating $s_{\hat{\beta}_i}$ due to small MSE) increases the chance of a Type I error by wrongly rejecting the H_0. On the other hand, underestimating t^* (e.g. by overestimating $s_{\hat{\beta}_i}$ due to large MSE) increases the chance of a Type II error by wrongly accepting H_0. Such incorrect hypothesis or significance tests can result from violating homoscedasticity, independence and/or normality of the residuals [13, 24].

Homoscedasticity means that the variance of the residuals is constant. If it is non-constant (heteroscedasticity) the estimated coefficients remain unbiased, but their estimated standard errors are biased. This can cause their confidence intervals and hypothesis test to be incorrect. Plots of residuals against predicted values or independent variables can aid in detecting non-constant variance, e.g. non-uniform spreads due to funneling or other patterns. Numerical tests are for example Levene's, Breusch-Pagan or White's test.

Non-independence of residuals can occur if observations close in time are more alike than observations more distant in time (time series data) or if observations within one group (e.g. belonging to the same workload) are more alike than observations from different groups (cross-section / clustered data). Non-independecse has the estimated coefficients remain unbiased, but their estimated standard errors biased. This can cause their confidence intervals and hypothesis tests to be incorrect. Plots of residuals e_i against e_{i-1} can aid in detecting non-independence of residuals, e.g. non-uniformly distributed residuals suggest autocorrelation [55]. Numerically the Durbin-Watson test can be performed to assess first-order autocorrelation. Roughly a value towards 0 indicates positive autocorrelation, a value towards 4 indicates negative autocorrelation and a value near 2 suggest no autocorrelation. The Durbin-Watson statistic:

$$d = \frac{\sum_{i=1}^{N} (e_i - e_{i-1})^2}{\sum_{i=1}^{N} e_i^2}$$

and critical values (table) can be used for such tests, however for large datasets of size N it can be approximated by a normal distribution with mean 2 and variance $\frac{4}{N}$ [43]. More general tests for autocorrelation exist, like Breusch-Godfrey.

Normality of residuals suggests residuals approximate random errors. Violation of the normality of the residuals matters to a lesser degree if the sample size is large. However, it might indicate the model does not fit the data well (e.g. due to a non-linear relationship). The estimated coefficients remain unbiased, however confidence intervals and hypothesis tests can be incorrect. A histogram of the residuals with an imposed normal curve can aid in detecting non-normality, e.g. the histogram does not follow the curve. Alternatively a normal probability plot of the normal distribution against the residual distribution can be used to detect non-normality, e.g. deviation from the straight (diagonal) line. Numerically the Kolmogorov-Smirnov or Shapiro-Wilk test can be applied.

To counter for non-normal residuals, the analysis should involve nonparameteric tests or one should employ a more advanced analysis like nonparametric regression. To counter for heteroscedasticity and autocorrelation, the analysis should involve estimates of standard errors that are more robust or one should employ a more advanced analysis like generalized least squares (GLS) regression [13, 41]. For heteroscedasticity only, it is possible to employ weighted least squares (WLS) regression [24, 55, 63]. Similar to OLS the aim is to estimate the coefficients β_i, however each case is given a weight w_i and one needs to minimize:

$$SSE_w = \sum_{i=1}^{N} w_i * (y_i - \hat{y}_i)^2 = \sum_{i=1}^{N} w_i * e_i^2$$

Note that OLS is a special case of WLS where $w_i = 1$. The advantage of WLS, however, is the ability to assign higher weights to cases with a small residual (and vice versa). This way heteroscedasticity is accounted for and even the effect of outliers can be downweighted. The difficulty is to determine the proper weights. SPSS has a weight estimation procedure that computes a weight for every case in the dataset. Along with the dependent and independent variables a weight variable is specified that relates to the change in variability of the dependent variable. The weight function is the reciprocal of the weight variable, say V, raised to the power of P:

$$\frac{1}{V^P}$$

By specifying a range of powers, SPSS iteratively computes the weights for each case and performs the weighted least squares regression. For every such regression model the value of a log-likelihood function is estimated. The power associated with the model that maximizes the log-likelihood value is considered the best. The weights w_i and regression model are known thereafter. Note that the difficulty is to determine the proper weight variable.

7.1.2 Validation

A good regression model does not only fit the training data well, but should provide reasonable predictions for unseen data too. The latter is expressed as a generalization error [59] and obtained from validating the regression model independent of the training data. Ideally both training and generalization error are low. A model with low training error and high generalization error suffers from model overfitting. High training and generalization errors indicate model underfitting. To this end the dataset for regression analysis is split into a training dataset and a test or validation data set. The former used for construction the regression model, the latter to independently validate the regression model.

Unfortunately SPSS is not too helpful in validating the regression models it derives. To this end, the training dataset is used to construct the model, while the validation data set is used to assess the generalization error, i.e. the performance of the model on unseen data. The use of independent training and validation data is employed often in assessing a model [42, 59, 64], e.g. for model selection in data mining. A straightforward approach is to divide data into a training and validation dataset as employed in this research, which in literature is referred to as the holdout method. The validation dataset is stratified to ensure each combination of retention and slide used during measurements is equally well represented. Although the data is randomly chosen, there is an apparent weakness in this approach: one might always end up with an unfortunate division of data causing either the training or validation dataset to be skewed by "bad" data, e.g. outliers, affecting validation results (positively or negatively).

A more powerful method is k-fold cross-validation which effectively performs k holdouts and averages the validation results. The data is partitioned into k datasets of (about) equal size. Each of those k partitions is used once as a validation set, while the other $k - 1$ datasets are used for training. The validation results are averaged to obtain the final validation result. Clearly, this method is less susceptible to the choice of partitions. Unfortunately SPSS provides no help in performing this kind of validation and performing it by hand takes considerable time. As such, the holdout method is the main method for validation of the regression models during the research. The validation data set contains 20% (10) of the measurements for each workload and is disjoint of the other 80% (40) of the measurements for each workload used for training. Moreover the training and validation dataset have each workload represented equally; stratified for retention-slide pairs used during evaluations.

Among the common functions to quantify the generalization error, are the root mean squared error ($RMSE_v$) and mean absolute error (MAE_v) [42, 59, 64]. The $RMSE_v$ takes the square root of the average squared error of the predicted against the observed values. The $RMSE_v$ of the folds are averaged. The

Processing time regression model Instantaneous approach: variance						
R^2	0.997					
Std. Error of the Estimate	0.011					
$F_{2,197}$	37258.317					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	0.002	0.003	0.001		1.655	insignificant
Retention	0.008	0.000	0.000	0.972	212.773	< 0.01
Slide	0.008	0.000	0.000	0.043	9.521	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.010					
MAE_v	0.008					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	No					

Table 7.1: processing time regression model for the instantaneous approach using a single-pass incremental formula. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

$RMSE_v$ is defined as follows:

$$RMSE_v = \sqrt{\frac{\sum_{i=1}^{N_{fold}} (y_i - \hat{y}_i)^2}{N_{fold}}} \text{ where } N_{fold} \text{ is the size of the fold or the holdout}$$

The MAE_v takes the average of the absolute error of the predicted against the observed values. The MAE_v of the folds are averaged. The MAE_v is defined as follows:

$$MAE_v = \frac{\sum_{i=1}^{N_{fold}} |y_i - \hat{y}_i|}{N_{fold}} \text{ where } N_{fold} \text{ is the size of the fold or the holdout}$$

The $RMSE_v$ tends to exaggerate the presence of outliers (and larger errors). Both will be used to quantify the generalization error of the models on the validation dataset.

7.2 Models: Variance

For each approach implementation evaluated for the bounded differential aggregate variance, a processing time and memory space usage model by means of (multiple) linear regression is derived. Each model is reported and discussed shortly. The plots used to evaluate the regression models are found in the appendix. See B.1 for plots regarding processing time models and B.2 regarding memory space usage models. The appendix also explains the interpretation of the plots.

7.2.1 Instantaneous

The processing time model for the instantaneous is summarized in table 7.1. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Heteroscedasticity is shown in graph B.1. It shows a funnel pattern that is clear in the plot of residuals against retention in B.2. The residuals are not normally distributed as shown in histogram B.3 and normal P-P plot in B.4. A Durbin-Watson statistic of 1.725 indicated by SPSS and lag plot B.17 confirm some positive autocorrelation. Weighted least squares regression, using e.g. retention as weight variable, was not fruitful (not shown).

The memory space usage model for the instantaneous is summarized in table 7.2. From the results in 6.3 and 6.4 it seems that retention affects the memory space usage the most. The regression model

Memory space usage regression model Instantaneous approach: variance						
R^2	0.999					
Std. Error of the Estimate	0.957					
$F_{2,197}$	108714.667					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	26.674	0.243	0.123		216.822	< 0.01
Retention	1.201	0.007	0.003	0.984	367.606	< 0.01
Slide	0.070	0.014	0.007	0.026	9.578	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.955					
MAE_v	0.882					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.2: Memory space usage regression model for the instantaneous approach. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ∘

suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in graph B.32. Residuals against retention in graph B.33 shows large variance for large retention. The residuals are not normally distributed as shown in histogram B.34 and normal P-P plot B.35. A Durbin-Watson statistic of 0.063 reported by SPSS and lag plot B.36 confirm the very strong positive autocorrelation. Weighted least squares regression is little fruitful as the spread is hard to account for.

7.2.2 Biased Reservoir Sampling

The processing time usage model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ is summarized in table 7.3. From the results and graphs 6.3 and 6.4 it seems that only slide affects the processing time. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the strength of the processing time model. Heteoscedasticity is shown in graph B.6. The residuals are not normally distributed as shown in histogram B.7 and normal P-P plot B.8. A Durbin-Watson statistic of 1.758 indicated by SPSS and lag plot B.9 confirm some positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

The memory space usage model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ is summarized in table 7.4. From the results and graphs 6.3 and 6.4 it seems that only slide affects the memory space usage. The regression model suggests a perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in graph B.37. The residuals are not normally distributed as shown in histogram B.38 and normal P-P plot B.39. A Durbin-Watson statistic of 0.244 indicated by SPSS and lag plot B.40 confirm the strong positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

The processing time model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$ is summarized in table 7.5. From the results and graphs 6.3 and 6.4 it seems that only slide affects the processing time. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ is more optimistic than the standard error of the estimate of the model on training data.

Processing time regression model						
Incremental approach using sampling ($\epsilon = 0.05, \delta = 0.0001$): variance						
R^2	0.999					
Std. Error of the Estimate	0.006					
$F_{1,198}$	189049.226					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	-0.001	0.001	0.001		-2.283	< 0.05
Slide	0.016	0.000	0.000	0.999	434.798	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.006					
MAE_v	0.004					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	No					

Table 7.3: processing time regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

Memory space usage regression model						
Incremental approach with sampling ($\epsilon = 0.05, \delta = 0.0001$): variance						
R^2	1.000					
Std. Error of the Estimate	0.042					
$F_{1,198}$	$2.238E^7$					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	25.899	0.008	0.004		6758.322	< 0.01
Slide	1.207	0.001	0.000	1.000	4730.933	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.037					
MAE_v	0.033					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.4: Memory space usage regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

Processing time regression model						
Incremental approach using sampling ($\epsilon = 0.01, \delta = 0.0001$): variance						
R^2	0.998					
Std. Error of the Estimate	0.009					
$F_{1,198}$	99348.417					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	0.000	0.001	0.001		-1.057	insignificant
Slide	0.017	0.000	0.000	0.999	315.196	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.004					
MAE_v	0.003					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Almost					
Normality	No					

Table 7.5: processing time regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

Some caution is required regarding the processing time model. Little heteroscedasticity is shown in graph B.10. The residuals are not normally distributed as shown in histogram B.11 and normal P-P B.12. A Durbin-Watson statistic of 2.049 indicated by SPSS and lag plot B.13 confirm slight positive autocorrelation. Weighted least squares regression is not viable as heteroscedasticity is not an issue, but rather normality of the residuals.

The memory space usage model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$ is summarized in table 7.6. From the results and graphs 6.3 and 6.4 it seems slide affects memory space usage the most. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data with $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in graph B.41, related to retention as shown in graph B.42 and by slide as shown in graph B.43. The residuals are not normally distributed as shown in histogram B.44 and normal P-P plot B.45. A Durbin-Watson statistic of 0.097 reported by SPSS and lag plot B.46 confirm the strong positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

7.2.3 Expiration List

The processing time model for the incremental approach using expiration lists is summarized in table 7.7. From the results and graphs 6.3 and 6.4 it seems that only slide affects the processing time. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ a little less optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Heteroscedasticity is shown in graph B.14. The residuals are not normally distributed as is shown in histogram B.15 and normal P-P plot B.16. A Durbin-Watson statistic of 1.893 indicated by SPSS and lag plot B.17 confirm some positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

The memory space usage model for the incremental approach using expiration lists is summarized in table 7.8. From the results and graphs 6.3 and 6.4 it seems both retention and slide affect memory space usage. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ close to the standard error of the estimate of the model on training data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in graph B.47, related to retention as shown in graph B.48 and by slide as shown in graph B.49. The residuals

Memory space usage regression model						
Incremental approach using sampling ($\epsilon = 0.01, \delta = 0.0001$): variance						
R^2	0.999					
Std. Error of the Estimate	0.569					
$F_{2,197}$	69604.706					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	27.003	0.145	0.073		369.050	< 0.01
Retention	-0.005	0.004	0.002	-0.008	-2.398	< 0.05
Slide	1.306	0.009	0.004	1.004	300.226	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.553					
MAE_v	0.413					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.6: Memory space usage regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ∘

Processing time regression model						
Incremental approach with expiration list: variance						
R^2	0.997					
Std. Error of the Estimate	0.016					
$F_{1,198}$	69658.085					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	-0.007	0.003	0.001		-4.626	< 0.01
Slide	0.025	0.001	0.000	0.999	263.928	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.023					
MAE_v	0.013					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	No					

Table 7.7: processing time regression model for the incremental approach using expiration lists. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ∘

Memory space usage regression model Incremental approach using expiration lists: variance						
R^2	0.964					
Std. Error of the Estimate	9.101					
$F_{2,197}$	2640.169					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	32.389	2.308	1.170		27.674	< 0.01
Retention	1.354	0.062	0.031	0.735	43.554	< 0.01
Slide	1.426	0.137	0.070	0.346	20.498	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	10.610					
MAE_v	7.791					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	Almost					

Table 7.8: Memory space usage regression model for the incremental approach using expiration lists. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ◦

are approximately normally distributed as shown in histogram B.50 and normal P-P plot B.51. A Durbin-Watson statistic of 1.575 reported by SPSS and lag plot B.52 suggest some positive autocorrelation.

Weighted regression is used to counter for heteroscedasticity. A weight estimation procedure using retention as the weight variable, next to being an independent variable, is performed yielding a power of 1. Hence each case is weighted by the reciprocal of the retention, e.g. $\frac{1}{1}$, $\frac{1}{15}$ or $\frac{1}{60}$ depending on the case). The weighted regression model for memory space usage is summarized in 7.9. A near perfect fit by an R^2 of 0.979 is suggested by the model. It should be noted that the R^2 of the OLS and WLS model cannot be compared straightforwardly as the latter is obtained on transformed data as is warned for in [63]. Validation is slightly more optimistic. Reduced heteroscedasticity can be observed from the plot of weighted residuals against weighted predicted values in graph B.53. Approximate normality is obvious from the histogram B.54 and normal probability plot B.55. Some (perhaps increased) positive autocorrelation is clear from the lag plot B.56.

7.2.4 Panes

The processing time model for the incremental approach using panes is summarized in table 7.10. From the results and graphs 6.3 and 6.4 it seems that mostly slide affects the processing time. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ close to the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Heteroscedasticity is shown in graph B.18. The residuals are not normally distributed as shown in histogram B.19 and normal P-P plot B.20. A Durbin-Watson statistic of 1.545 indicated by SPSS and lag plot B.21 confirm some positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

The memory space usage model for the incremental approach using panes is summarized in table 7.11. From the results and graphs 6.3 and 6.4 it seems that slide affects the memory space usage the most. The regression model suggests a perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in graph B.57, related to retention in graph B.58 and related to slide in graph B.59. The residuals are not normally distributed as shown in histogram B.60 and normal P-P B.61. A Durbin-Watson statistic of 0.899 indicated by SPSS and lag plot B.62 confirm the positive autocorrelation. Weighted least squares regression is little fruitful as the spread is hard to account for (not shown).

Memory space usage weighted regression model Incremental approach with expiration list: variance						
R^2	0.979					
Std. Error of the Estimate	1.542					
$F_{2,197}$	4617.263					
p	< 0.01					
Weight	$\frac{1}{Retention}$					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	29.463	0.495	0.251		117.383	< 0.01
Retention	1.465	0.054	0.027	0.815	53.574	< 0.01
Slide	1.277	0.173	0.088	0.221	14.539	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	10.473					
MAE_v	7.413					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Almost					
Normality	Almost					

Table 7.9: Memory space usage regression model for the incremental approach using expiration lists from weighted least squares regression. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ○

Processing time regression model Incremental approach with panes: variance						
R^2	1.000					
Std. Error of the Estimate	0.002					
$F_{2,197}$	654168.094					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	0.000	0.001	0.000		0.171	insignificant
Retention	0.000	0.000	0.000	0.003	3.001	< 0.01
Slide	0.016	0.000	0.000	0.998	914.209	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.003					
MAE_v	0.001					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	No					

Table 7.10: processing time regression model for the incremental approach using panes. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ○

Memory space usage regression mode						
Incremental approach with panes: variance						
R^2	1.000					
Std. Error of the Estimate	0.156					
$F_{2,197}$	806480.420					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	25.842	0.039	0.020		1298.459	< 0.01
Retention	0.002	0.001	0.001	0.003	3.276	< 0.01
Slide	1.201	0.003	0.001	0.998	1015.108	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.121					
MAE_v	0.105					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.11: Memory space usage regression model for the incremental approach using panes. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ◦

7.2.5 Variance Histogram

The processing time model for the incremental approach using variance histograms with $\epsilon = 0.05$ is summarized in table 7.12. From the results and graphs 6.3 and 6.4 it seems that only slide affects the processing time. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ slightly less optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Heteroscedasticity is shown in graph B.22. The residuals are not normally distributed as shown in histogram B.23 and normal P-P plot B.24. A Durbin-Watson statistic of 1.011 indicated by SPSS and lag plot B.25 confirm positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

The memory space usage model for the incremental approach using variance histograms with $\epsilon = 0.05$ is summarized in table 7.13. From the results and graphs 6.3 and 6.4 it seems that slide affects the memory space usage the most. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in graph B.63, related to retention in graph B.64 and related to slide in graph B.65. The residuals are not normally distributed as shown in histogram B.66 and a normal P-P plot B.67. A Durbin-Watson statistic of 2.438 indicated by SPSS and lag plot B.68 confirm hardly any autocorrelation.

The funnel patterns of the residuals against slide in graph B.64 suggest that slide has the strongest impact on heteroscedasticity as its values increase. A weight estimation procedure using slide as the weight variable, next to being an independent variable, is performed yielding a power of 1.2. The weighted regression model for memory space usage is summarized in table 7.14. A near perfect fit by an R^2. It should be noted that the R^2 of the OLS and WLS model cannot be compared straightforwardly as the latter is obtained on transformed data as is warned for in [63]. Validation shows similar values as for the unweighted model. Reduced heteroscedasticity can be observed in graph B.69. Approximate normality is obvious from the histogram B.70 and normal probability plot B.71. Some (perhaps reduced) negative autocorrelation is clear from the lag plot B.72.

The processing time model for the incremental approach using variance histograms with $\epsilon = 0.01$ is summarized in table 7.15. From the results and graphs 6.3 and 6.4 it seems that mostly slide affects the processing time. The regression model suggests a near perfect fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows a reasonably good

Processing time regression model						
Incremental approach with histogram ($\epsilon = 0.05$): variance						
R^2	0.999					
Std. Error of the Estimate	0.015					
$F_{1,198}$	189633.919					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	-0.006	0.003	0.001		-5.119	< 0.01
Slide	0.037	0.001	0.000	0.999	435.470	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.019					
MAE_v	0.013					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.12: Processing time regression model for the incremental approach using variance histograms with $\epsilon = 0.05$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

Memory space usage regression model						
Incremental approach with histogram: variance						
R^2	0.972					
Std. Error of the Estimate	5.761					
$F_{2,197}$	3433.604					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	28.810	1.461	0.741		38.889	< 0.01
Retention	0.055	0.039	0.020	0.042	2.815	< 0.01
Slide	2.846	0.087	0.044	0.960	64.639	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	5.833					
MAE_v	3.824					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	Almost					

Table 7.13: Memory space usage regression model for the incremental approach using variance histograms with $\epsilon = 0.05$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

Memory space usage weighted regression model **Incremental approach with histogram ($\epsilon = 0.05$): variance**						
R^2	0.948					
Std. Error of the Estimate	1.473469					
$F_{2,197}$	1780.017					
p	< 0.01					
Weight	$\frac{1}{Slide^{1.2}}$					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	28.864	0.386	0.196		147.366	< 0.01
Retention	0.037	0.0105	0.005	0.116	7.013	< 0.01
Slide	2.919	0.101	0.051	0.947	57.178	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	5.878					
MAE_v	3.835					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Almost					
Normality	Almost					

Table 7.14: Memory space usage regression model for the incremental approach using variance histograms with $\epsilon = 0.05$ from weighted least squares regression. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. $\circ$

performance on unseen data and $RMSE_v$ slightly more optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Heteroscedasticity is shown in graph B.26, related to slide in graph B.28 and related to retention in graph B.27. The residuals are approximately normally distributed as shown in histogram B.29 and normal P-P plot B.30. A Durbin-Watson statistic of 1.607 indicated by SPSS and lag plot B.31 confirm some positive autocorrelation. Weighted least squares regression was not fruitful (not shown).

The memory space usage model for the incremental approach using variance histograms with $\epsilon = 0.01$ is summarized in table 7.16. from the results and graphs 6.3 and 6.4 it seems that slide affects the memory space usage the most. The regression model suggests a good fit by the R^2 coefficient of determination and by the F-test a statistically significant model as $p < 0.01$. Validation shows $RMSE_v$ error close to the standard error of the estimate of the model on training data.

Some caution is required regarding the memory space usage model. Heteroscedasticity is shown in grpah B.73, related to retention in graph B.74 and related to slide in graph B.75. The residuals are not normally distributed as shown in histogram B.76 and normal P-P plot B.77. A Durbin-Watson statistic of 0.905 reported by SPSS and lag plot B.68 confirm the positive autocorrelation.

The funnel patterns of the residuals against retention in graph B.74 suggest that retention has the strongest impact on heteroscedasticity as its values increase. A weight estimation procedure using retention as the weight variable, next to being an independent variable, is performed yielding a power of 1.2. The weighted regression model for memory space usage is summarized in 7.17. A near perfect fit by an R^2. It should be noted that the R^2 of the OLS and WLS model cannot be compared straightforwardly as the latter is obtained on transformed data as is warned for in [63]. Validation shows similar values as for the unweighted model. Reduced heteroscedasticity can be observed in graph B.79. Slight normality is obvious in histogram B.80 and normal P-P plot B.81. Some positive autocorrelation is shown in lag plot B.82.

7.3 Models: Quantiles

For each approach implementation evaluated for the unbounded non-differential aggregate quantiles, a processing time and memory space model by means of (multiple) linear regression is derived. For each approach, the models are reported and discussed shortly. The plots used to evaluate the regression models are found in the appendix. See B.3 for plots regarding processing time models and B.4 for plots regarding memory space usage models. The appendix also explains the interpretation of the plots.

Processing time regression model						
Incremental approach with histogram ($\epsilon = 0.01$): variance						
R^2	0.996					
Std. Error of the Estimate	0.032					
$F_{2,197}$	23445.642					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	0.000	0.009	0.004		-0.107	insignificant
Retention	0.002	0.001	0.000	0.124	21.518	< 0.01
Slide	0.039	0.001	0.000	0.919	159.669	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.027					
MAE_v	0.020					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	Almost					

Table 7.15: Processing time regression model for the incremental approach using variance histograms with $\epsilon = 0.01$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ○

Memory space usage regression model						
Incremental approach with histogram ($\epsilon = 0.01$): variance						
R^2	0.886					
Std. Error of the Estimate	14.330					
$F_{2,197}$	777.278					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	34.253	3.634	1.843		18.586	< 0.01
Retention	1.137	0.097	0.049	0.693	23.224	< 0.01
Slide	1.271	0.216	0.110	0.346	11.607	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	15.297					
MAE_v	11.866					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.16: Memory space usage regression model for the incremental approach using variance histograms with $\epsilon = 0.01$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ○

Memory space usage weighted regression model						
Incremental approach with histogram ($\epsilon = 0.01$): variance						
R^2	0.938					
Std. Error of the Estimate	1.637					
$F_{2,197}$	1502.487					
p	< 0.01					
Weight	$\frac{1}{Retention^{1.2}}$					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	29.428	0.553	0.280		105.094	< 0.01
Retention	1.350	0.083	0.042	0.830	32.001	< 0.01
Slide	0.953	0.274	0.139	0.179	6.882	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	16.506					
MAE_v	11.538					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Slight					
Normality	Slight					

Table 7.17: Memory space usage regression model for the incremental approach using variance histograms with $\epsilon = 0.01$ from weighted least squares regression. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI - 95\%$. ∘

7.3.1 Instantaneous

The processing time model for the instantaneous approach is summarized in table 7.18. From the results in 6.3.1 it is clear slide has no significant impact. It is excluded from the regression model (not shown). The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonably good performance on unseen data and an $RMSE_v$ similar to the standard error of the estimate of the model on the training data.

Some caution is required regarding the processing time model. The presence of heteroscedasticity is shown in graph B.83. The residuals are not normally distributed as is clear from the histogram B.84 and the normal P-P plot B.85. A Durbin-Watson statistic of 0.814 reported by SPSS and lag plot B.86 suggest positive autocorrelation. A processing time weighted model did not improve the results (not shown).

The memory space usage model for the instantaneous approach is summarized in table 7.19. From the results in 6.3.1 it is clear slide has no significant impact. It is excluded from the regression model (inclusion would show its insignificance). The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonably good performance on unseen data and a $RMSE_v$ similar to the standard error of the estimate of the model on the training data.

Some caution is required regarding the memory space usage model. The presence of heteroscedasticity is shown in graph B.111. The residuals are not normally distributed as is clear from the histogram B.112 and the normal P-P plot B.113. A Durbin-Watson statistic of 2.510 reported by SPSS and lag plot B.114 suggest minor negative autocorrelation.

Given the heteroscedasticity, a weighted regression model is tried. The memory space usage weighted model for the instantaneous approach is summarized in table 7.20. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonably good performance on unseen data, though comparable to the unweighted model. Reduced heteroscedasticity in graph B.115 and improved normality in histogram B.116 or normal P-P plot B.117 can be observed. Autocorrelation is similar as shown in lag plot B.118.

7.3.2 Biased Reservoir Sampling

The processing time model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ is summarized in table 7.21. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation

Processing time regression model Instantaneous approach: quantiles						
R^2	0.992					
Std. Error of the Estimate	0.116					
$F_{1,198}$	23456.343					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	-0.145	0.030	0.015		-9.681	< 0.01
Retention	0.049	0.001	0.000	0.996	153.155	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.104					
MAE_v	0.089					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	No					

Table 7.18: Processing time regression model for the instantaneous approach. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ◦

Memory space usage regression model Instantaneous approach: quantiles						
R^2	0.958					
Std. Error of the Estimate	11.476					
$F_{1,198}$	4572.447					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	27.544	2.906	1.474		18.690	< 0.01
Retention	2.122	0.062	0.031	0.979	67.620	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	12.780					
MAE_v	8.963					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	No					

Table 7.19: Memory space usage regression model for the instantaneous approach. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ◦

Memory space usage weighted regression model						
Instantaneous approach: quantiles						
R^2	0.984					
Std. Error of the Estimate	0.113					
$F_{1,198}$	12516.954					
p	< 0.01					
Weight	$\frac{1}{Retention}$					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	26.889	0.053	0.027		1004.061	< 0.01
Retention	2.163	0.038	0.019	0.992	111.879	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	12.696					
MAE_v	9.051					
Assumptions (Residuals)						
Homoscedasticity	Slight					
Independence	Almost					
Normality	Slight					

Table 7.20: Memory space usage weighted regression model for the instantaneous approach. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ∘

shows a reasonably good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. The presence of heteroscedasticity is shown in graph B.87. The residuals are approximately normally distributed as is clear from the histogram B.88 and the normal P-P plot B.89. A Durbin-Watson statistic of 1.356 reported by SPSS and lag plot B.90 suggest positive autocorrelation. A processing time weighted model did not improve the results (not shown).

The memory space usage model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$ is summarized in table 7.22. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows performance is reasonably good on unseen data and $RMSE_v$ similar to the standard error of the estimate from the model on the training data.

Caution is required regarding the memory space usage model. The presence of heteroscedasticity is shown in graph B.119. The residuals are approximately normally distributed as is clear from the histogram B.120 and the normal P-P plot B.121. A Durbin-Watson statistic of 0.381 reported by SPSS and lag plot B.122 suggest positive autocorrelation.

By the heteroscedasticity is makes sense to try weighted least squares regression using slide as a weight variable. The weight estimation yields a power of -1.4. The weighted regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows performance is reasonably good on unseen data and $RMSE_v$ more optimistic than the standard error of the estimate from the model on the training data. Performance is not as good as the unweighted model. Heteroscedasticity is reduced as shown in graph B.123. The residuals are approximately normally distributed as is clear from the histogram B.124 and the normal P-P plot B.125. The lag plot B.126 suggest reduced positive autocorrelation.

The processing time model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$ is summarized in table 7.24. From the results in 6.3.1 retention is insignificant. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ more optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. The presence of heteroscedasticity is shown in graph B.91. The residuals are somewhat approximating a normal distribution as is clear from the histogram B.92 and the normal P-P plot B.93. A Durbin-Watson statistic of 2.228 reported by SPSS and lag plot B.94 suggests hardly any autocorrelation.

By the heteroscedasticity is makes sense to try weighted least squares regression using slide as a weight variable. The weight estimation yields a power of 1.1. The weighted regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-

Processing time regression model **Incremental approach with biased reservoir sampling ($\epsilon = 0.05, \delta = 0.0001$): quantiles**						
R^2	0.999					
Std. Error of the Estimate	0.022					
$F_{2,197}$	71023.516					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	0.002	0.003	0.003		0.814	insignificant!
Retention	0.000	0.000	0.000	-0.007	-2.092	< 0.05
Slide	0.051	0.000	0.000	1.003	303.074	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.025					
MAE_v	0.013					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	Almost					

Table 7.21: Processing time regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ◦

Memory space usage regression model **Incremental approach with biased reservoir sampling ($\epsilon = 0.05, \delta = 0.0001$): quantiles**						
R^2	0.988					
Std. Error of the Estimate	1.504					
$F_{2,197}$	8455.411					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	30.432	0.382	0.193		157.317	< 0.01
Retention	-0.063	0.010	0.005	-0.117	-12.220	< 0.01
Slide	1.276	0.023	0.011	1.060	110.999	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	1.535					
MAE_v	1.152					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	Almost					

Table 7.22: Memory space usage regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ◦

Memory space usage weighted regression model **Incremental approach with biased reservoir sampling ($\epsilon = 0.05, \delta = 0.0001$): quantiles**						
R^2	0.998					
Std. Error of the Estimate	2.243					
$F_{2,197}$	52052.494					
p	< 0.01					
$Weight$	$\frac{1}{Slide^{-1.4}}$					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	30.292	0.566	0.287		105.560	< 0.01
Retention	-0.034	0.010	0.005	-0.022	-6.557	< 0.01
Slide	1.212	0.008	0.004	1.007	303.089	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	1.722					
MAE_v	1.327					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Slight					
Normality	Almost					

Table 7.23: Memory space usage weighted regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. $\circ$

Processing time regression model **Incremental approach with biased reservoir sampling ($\epsilon = 0.01, \delta = 0.0001$): quantiles**						
R^2	0.999					
Std. Error of the Estimate	0.050					
$F_{1,198}$	136595.362					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	0.033	0.010	0.005		7.099	< 0.01
Slide	0.113	0.001	0.000	0.999	369.588	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.034					
MAE_v	0.023					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Almost					
Normality	Some					

Table 7.24: Processing time regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. $\circ$

Processing time weighted regression model						
Incremental approach with biased reservoir sampling ($\epsilon = 0.01, \delta = 0.0001$): quantiles						
R^2	0.997					
Std. Error of the Estimate	0.014					
$F_{1,198}$	72755.537					
p	< 0.01					
$Weight$	$\frac{1}{Slide^{-1.1}}$					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	0.032	0.003	0.001		22.544	< 0.01
Slide	0.113	0.001	0.000	0.999	269.732	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.034					
MAE_v	0.023					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Almost					
Normality	Almost					

Table 7.25: Processing time weighted regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ∘

test as $p < 0.01$. Validation shows performance is reasonably good on unseen data and $RMSE_v$ less optimistic than the standard error of the estimate from the model on the training data. Performance is similar to the unweighted model. Heteroscedasticity is reduced as shown in graph B.95. The residuals are approximately normally distributed as is clear from the histogram B.96 and the normal P-P plot B.97. The lag plot B.98 suggests hardly any autocorrelation. The weighted model hardly violates the assumptions and has yields a model similar to the unweighted model.

The memory space usage model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$ is summarized in table 7.26. The regression model suggests a low fit by the R^2 coefficient of determination, though a statistically significant model by the F-test as $p < 0.01$. Validation shows performance similar to the standard error of the estimate from the model on the training data. Performance of the model is low.

Caution is required regarding the memory space usage model. Heteroscedasticity is not an issue as shown in graph B.127. The residuals are approximately normally distributed as is clear from the histogram B.128 and the normal P-P plot B.129. A Durbin-Watson statistic of 0.691 reported by SPSS and lag plot B.130 suggest positive autocorrelation. As heteroscedasticity is not an issue, weighted least squares regression seems no viable alternative (not shown).

7.3.3 Expiration List

The processing time model for the incremental approach using expiration lists is summarized in table 7.27. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonably good performance on unseen data and $RMSE_v$ more optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. The presence of heteroscedasticity is shown in graph B.99. The residuals are somewhat normally distributed as is clear from the histogram B.100 and the normal P-P plot B.101. A Durbin-Watson statistic of 2.044 reported by SPSS and lag plot B.102 suggest some positive autocorrelation. A processing time weighted model did not improve the results (not shown).

The memory space usage model for the incremental approach using expiration lists is summarized in table 7.28. From the results in 6.3.1 it is clear slide has no significant impact. It is excluded from the regression model (inclusion would show its insignificance). The regression model suggests a low fit by the R^2 coefficient of determination though a statistically significant model by the F-test as $p < 0.01$. Validation shows a $RMSE_v$ similar to the standard error of the estimate from the model on the training data. The performance not so good.

Caution is required regarding the memory space usage model. The presence of heteroscedasticity is

Memory space usage regression model Incremental approach with biased reservoir sampling ($\epsilon = 0.01, \delta = 0.0001$): quantiles						
R^2	0.505					
Std. Error of the Estimate	6.052					
$F_{2,197}$	100.454					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	74.234	1.535	0.778		95.376	< 0.01
Retention	-0.082	0.041	0.021	-0.247	-3.952	< 0.01
Slide	0.614	0.091	0.046	0.831	13.268	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	6.241					
MAE_v	4.776					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	No					
Normality	Almost					

Table 7.26: Memory space usage regression model for the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ○

Processing time regression model Incremental approach with expiration list: quantiles						
R^2	0.986					
Std. Error of the Estimate	0.292					
$F_{2,197}$	6914.551					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	-0.117	0.074	0.037		-3.125	< 0.01
Retention	0.005	0.002	0.001	0.052	4.901	< 0.01
Slide	0.203	0.005	0.002	0.961	91.159	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.104					
MAE_v	0.089					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	Some					
Normality	Some					

Table 7.27: Processing time regression model for the incremental approach using expiration lists. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ○

Memory space usage regression model						
Incremental approach with expiration list: quantiles						
R^2	0.615					
Std. Error of the Estimate	66.451					
$F_{1,198}$	316.097					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	65.269	16.829	8.534		7.648	< 0.01
Retention	3.231	0.358	0.182	0.784	17.779	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	67.048					
MAE_v	53.102					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	Slight					

Table 7.28: Memory space usage regression model for the incremental approach using expiration lists. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. $\circ$

shown in graph B.131. The residuals are not normally distributed as is clear from the histogram B.132 and the normal P-P plot B.133. A Durbin-Watson statistic of 0.812 reported by SPSS and lag plot B.134 suggest positive autocorrelation. A memory space usage weighted model did not improve the results (not shown).

7.3.4 Arasu-Manku (Fixed Sketch)

The processing time model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$ is summarized in table 7.29. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows less performance on unseen data. The $RMSE_v$ is less optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Slight heteroscedasticity (apart from some outliers) is shown in graph B.103. The residuals are slightly normally distributed as is clear from the histogram B.104 though less obvious from the normal P-P plot B.105. A Durbin-Watson statistic of 1.314 reported by SPSS and lag plot B.106 suggests positive autocorrelation. Given the slight heteroscedasticity, a processing time weighted model will not improve the results (not shown).

The memory space usage model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$ is summarized in table 7.30. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonable good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Caution is required regarding the memory space usage model. The presence of heteroscedasticity is shown in graph B.135. The residuals slightly approximate a normal distribution as is clear from the histogram B.136 and the normal P-P plot B.137. A Durbin-Watson statistic of 0.582 reported by SPSS and lag plot B.138 suggest positive autocorrelation. A memory space usage weighted model did not improve the results (not shown).

The processing time model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$ is summarized in table 7.31. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a better performance on unseen data. The $RMSE_v$ is far more optimistic than the standard error of the estimate of the model on training data.

Some caution is required regarding the processing time model. Slight heteroscedasticity (apart from some outliers) is shown in graph B.107. The residuals are not normally distributed as shown in histogram B.108 and normal P-P plot B.109. A Durbin-Watson statistic of 1.559 reported by SPSS and lag plot B.110 suggests some positive autocorrelation. Given the slight heteroscedasticity, a processing time

Processing time regression model **Incremental approach with Arasu-Manku fixed sketches ($\epsilon = 0.05$): quantiles**						
R^2	0.997					
Std. Error of the Estimate	0.091					
$F_{2,197}$	37855.792					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	0.003	0.074	0.011		0.306	insignificant!
Retention	0.001	0.002	0.000	0.012	2.730	< 0.01
Slide	0.137	0.005	0.001	0.991	218.708	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.148					
MAE_v	0.064					
Assumptions (Residuals)						
Homoscedasticity	Some					
Independence	Slight					
Normality	Slight					

Table 7.29: Processing time regression model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ◦

Memory space usage regression model **Incremental approach with Arasu-Manku fixed sketch ($\epsilon = 0.05$): quantiles**						
R^2	0.994					
Std. Error of the Estimate	1.038					
$F_{2,197}$	17109.728					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{197}	p
(Constant)	27.842	0.264	0.134		208.552	< 0.01
Retention	0.027	0.358	0.004	0.052	7.730	< 0.01
Slide	1.138	0.358	0.008	0.965	143.382	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.904					
MAE_v	0.736					
Assumptions (Residuals)						
Homoscedasticity	No					
Independence	No					
Normality	Slight					

Table 7.30: Memory space usage regression model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ◦

Processing time regression model						
Incremental approach with Arasu-Manku fixed sketches ($\epsilon = 0.01$): quantiles						
R^2	0.991					
Std. Error of the Estimate	0.212					
$F_{1,198}$	23178.931					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	-0.024	0.040	0.020		-1.248	insignificant!
Slide	0.198	0.003	0.001	0.996	152.246	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	0.071					
MAE_v	0.051					
Assumptions (Residuals)						
Homoscedasticity	Some					
Independence	Some					
Normality	No					

Table 7.31: Processing time regression model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ∘

weighted model will not improve the results (not shown).

The memory space usage model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$ is summarized in table 7.32. From the results in 6.3.1 it is clear retention has no significant impact. The regression model suggests a near perfect fit by the R^2 coefficient of determination and a statistically significant model by the F-test as $p < 0.01$. Validation shows a reasonable good performance on unseen data and $RMSE_v$ similar to the standard error of the estimate of the model on training data.

Caution is required regarding the memory space usage model. Heteroscedasticity is hardly present shown in graph B.139. The residuals approximate a normal distribution as is clear from the histogram B.140 and the normal P-P plot B.141. A Durbin-Watson statistic of 1.623 reported by SPSS suggest some positive autocorellation though lag plot B.142 suggest hardly any autocorrelation. A memory space usage weighted model will not improve the results as heteroscedasticity is not a problem (not shown).

7.4 Limitations

During the regression analysis the assumptions of homoscedasticity, independence and normality of the residuals are often violated. This causes the statistical strength of the models to be weak. The $\hat{\beta}_i$ cannot be considered BLUE as its standard error is biased. This can cause standard errors, confidence intervals and hypothesis tests to be incorrect. With respect to the sum of squared errors (SSE) the coefficients might not be the best.

A limitation of the models stems from the measurements on which they are constructed. The limited variety in retention and slide reduced empirical evaluation efforts significantly at the expense of a more dense representation of retention and slide in the measurements. As a result, the models might fit the measurements well, but not generalize well to other choices of retention and slide (negatively impacting interpolation and extrapolation). This is a weakness of the setup that can be accounted for by performing a more extensive empirical evaluation. Whether this mitigates the violations of homoscedasticity, independence and normality of the residuals is not obvious. Another related limitation stems for the batch size B being equal to the slide S during measurements. As argued for in chapter 5, it is a consequence the way user defined aggregates are implemented in StreamInsight. Because for each window output is generated, the incremental approach outputs more frequently than the instantaneous approach if multiple batches / small windows make up a slide. It would break output semantics. As a result the regression models do not reflect arbitrary choices of batch sizes.

Some common causes of homoscedasticity are [41]:

- Issues in data collection.
- The presence of outliers.

Memory space usage regression model						
Incremental approach with Arasu-Manku fixed sketch ($\epsilon = 0.01$): quantiles						
R^2	0.992					
Std. Error of the Estimate	1.259					
$F_{1,198}$	25070.401					
p	< 0.01					
Coefficients						
Predictor	B	CI-95%	Std. Error	β	t_{198}	p
(Constant)	32.389	0.228	0.116		279.706	< 0.01
Slide	1.221	0.016	0.008	0.996	158.336	< 0.01
Validation, 20% holdout by stratified random sample						
$RMSE_v$	1.398					
MAE_v	1.143					
Assumptions (Residuals)						
Homoscedasticity	Almost					
Independence	Almost					
Normality	Almost					

Table 7.32: Memory space usage regression model for the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. B is the unstandardized and β the standardized coefficient. CI-95% is the 95% confidence interval for $B \pm CI$. ○

- Nonlinear relationships between the independent and dependent variables
- Omitted and possibly hidden variable(s).

Data collection is improved by a more extensive empirical evaluation; setup improvement. More effort might be put in the removal of outliers; analysis improvement. Nonlinear relationships are hard to account for. Based on the computational complexity of the approaches (e.g. variance histograms of section 4.6 in the incremental approach), non-linear relationships in the measurements are inevitable. Transformations might be applied to establish a linear form relationship, however attempts to do so were not fruitful (and are not reported). More advanced analysis, e.g. nonlinear regression, might be helpful to deal with nonlinear relationships. The existence of hidden variable(s), omitted in the model, cannot be excluded. Clearly their identification is non-trivial and requires further investigation.

Some common causes of autocorrelation are [41]:

- Manipulation of raw empirical data (e.g. smoothing by averaging).
- Non-linear relationships between the independent and dependent variables
- Omitted and possibly hidden variable(s).

The latter two relate to the causes of heteroscedasticity discussed above. The former can be mitigated by working with raw measurements. In this research limited manipulation has taken place on the raw data: changing the resolution (seconds to milliseconds instead of microseconds and nanoseconds; megabytes to kilobytes instead of bytes) and subsampling. However, attempts at working with the raw data have not shown benefits (and are not reported), hence does not seem to cause autocorrelation. Note that a procedure to transform data to remove the effect of first order autocorrelation is described in literature [24], though its application is tedious in SPSS and consequently not performed.

Non-normality of the residuals likely shares the causes discussed for heteroscedasticity and autocorrelation above. Besides they need not have a normal distribution. However, in that case other tests are required for statistical inference. Nonparametric tests and regression might be viable. In short, accounting for heteroscedasticity, autocorrelation and non-normality of the residuals is not an easy task. Nonlinear relationships and omitted variables are challenging. Other than improving data collection by a more extensive empirical evaluation with larger variety in retention and slice, a more advanced analysis seems beneficial. Generalized least squares (GLS), nonparametric and nonlinear regression provide robustness for (some of) the violated assumptions [13, 41].

The validation of the regression models shows the generalization error ($RMSE_v$) close to the standard error of the estimate ($RMSE$) of the regression models. From this it seems the standard error of the estimate of the regression models is reliable and the models perform conform expectation on unseen data. For example the standard error of the estimate for the processing time regression model of the instantaneous approach for variance in table 7.1 shows a standard error of the estimate ($RMSE$) of 0.011 and a $RMSE_v$ of 0.010, hence the latter is conform expectation with respect to the former. It should be noted that the validation data is based on the same measurements as the data on which the models are constructed. Hence the same issue of limited variety in retention and slide emerges. The validation could be improved by a more extensive empirical evaluation yielding better validation data.

7.5 Wrap-up

This chapter has described regression analysis performed on the empirical results of chapter 6 using the statistical software SPSS. The regression models are used as empirical approximations of the theoretical model in chapter 3. Their use is elaborated on in chapter 8. The regression analysis and assumptions, regarding the statistical strength of the regression analysis, are discussed. For each approach (implementation) evaluated, a 20% holdout of empirical results is used to independently validate the models derived from the other 80% (training data) of the empirical results. Most models violate the assumptions of homoscedasticity (non-constant variance), normality and independence of the residuals. This causes the statistical strength of the models to be weak. Hence estimators of the coefficients of the regression models should not be considered BLUE, i.e. they can still be unbiased but do not have the minimum variance and hence are not the best coefficients with respect to the ordinary least squares (minimization of SSE). Validation of the models shows an error similar to the standard error of the estimate for the models from regression analysis. More advanced regression analysis is required that employ robust estimates of standard errors, nonparametric regression, generalized least squares (GLS) regression and non-linear regression. As batch size is equal to the slide during measurements (see chapter 5), the regression models do not reflect arbitrary choices of batch size.

Chapter 8

Design Space and Guidelines

This chapter discusses the design space that can be used to compare the performance of the approaches and determine which performs best. Its purpose is to answer the main research question from the perspective of performance; the best performing approach determines how to compute an aggregate over sliding windows. The theoretical model introduced in chapter 3 forms the basis. It is augmented with guidelines derived from the aggregates in chapter 3, synopsis structures in chapter 4, empirical results in chapter 6 and regression models in chapter 7.

In section 8.1 the use of the theoretical model is discussed. It starts by using it as the scalability model by instantiating it with computational complexity. An example demonstrates it. Next the theoretical model is instantiated with empirical approximations. A procedure for selecting the regression models, used for empirical approximations, and a comparison of performance of the approaches is given. Again an example demonstrates it. The chapter proceeds with a discussion of the generalization of and limitations on the design space in section 8.2. Finally in section 8.3 an overview of the guidelines from the previous chapters is given.

8.1 Design Space

The design space is to determine which approach seems best to compute an aggregate, by comparing performance of the approaches for given output semantics: input, aggregate, accuracy requirements, retention and slide. By using the theoretical model of chapter 3 the design space can be used to compare the approaches in two ways: by scalability through computational complexity and by empirical approximations.

Regarding the scalability model, approaches can be compared in the following way for a given aggregate (analogous to what is done in 3.2):

1. Compare processing time:
   - (a) Determine the time complexity of the algorithm to compute the aggregate under the instantaneous approach; an instantiation of $T(R)$.
   - (b) Determine the time complexity of the update and query / lookup time of the synopsis structure for the aggregate under the incremental approach; an instantiation of $T_u(R)$ and $T_q(R)$ respectively in $T(R, S, B)$.
   - (c) Use the ratios 3.2.1, 3.2.2 or 3.2.3 in the limit of R to infinity to determine how the approaches relate to each other in terms of processing time.
   - (d) Use the limits of S and B to 1 or R to determine the consequences of small slides and batches or large slides and batches for the incremental approach.
2. Compare memory space usage:
   - (a) Determine the space complexity of the large window and algorithm to compute the aggregate under the instantaneous approach; an instantiation of $S(R)$.
   - (b) Determine the space complexity of the batch / small window and synopsis structure for the aggregate under the incremental approach; an instantiation of $S(B, R)$.

(c) Use the ratios (as discussed for comparing processing time) to determine how the approaches relate to each other in terms of memory space usage.

(d) Use the limit of B to 1 or R to determine the consequences of small and large slides for the incremental approach.

By the ratios it can be shown whether one approach bounds another from below $\Omega(\cdot)$, from above $O(\cdot)$ or are in order $\Theta(\cdot)$. Note that in section 3.5 the time and space complexity of the instantaneous approach for the selected aggregates is mentioned. In chapter 4 the update time, query time and maintenance space complexity of the synopsis structures is mentioned, taking into account the selected aggregates. An overview of the complexities is given in table 8.1. An example will demonstrate the scalability model.

Example 8.1.1. In the instantaneous approach computing the quantiles takes time $T(R) \sim O(R * \log R + R)$ to sort the elements and determine all quantiles of interest over the sorted elements. Clearly it requires space $S(R) \sim O(2 * R) \sim O(R)$ to keep the large window of elements and the sorted elements. In the incremental approach suppose elements are kept value ordered in a red-black tree and time ordered in a linked list. The red-black tree has update time $O(\log R)$ for inserts and removals, the linked list has time $O(1)$ for appends and removals in front. Each new element inserted removes an expired element and requires $T_u(R) = O(2 + 2 * \log R) \sim O(\log R)$. To determine the quantiles a pass over the red-black tree is required in time $O(R)$. Space $O(2 * R + B) \sim O(R)$ is required to store the red-black tree, the linked list and the batch / small window.

The comparison is as follows:

$$\lim_{R\to\infty} \frac{T(R,S,B)}{T(R)} = \lim_{R\to\infty} \frac{S * \log R + R}{R * \log R + R} = \lim_{R\to\infty} \frac{\frac{S}{R} + \frac{1}{\log R}}{1 + \frac{1}{\log R}} = 0$$

This means $T(R,S,B) \in O(T(R))$, so the instantaneous approach bounds processing time from above for some constant S as the retention gets sufficiently large. Moreover $T(R,S,B) \notin \Theta(T(R))$, hence they do not grow at the same rate. It can be observed from the empirical results in chapter 6 in section 6.3.1 too. ◇

Next to scalability, the performance can be approximated empirically. The regression models in chapter 7 for processing time approximate $T(R)$ and $T(R,S,B)$ for the instantaneous and incremental approaches respectively. The regression models for memory space usage approximate $S(R)$ and $S(B,R)$ for the instantaneous and incremental approaches respectively. Note that in chapter 5 it is argued that batch size B is equal to slide S during measurements, which most likely affect memory space usage $S(B,R)$ in the incremental approach. It seems insignificant to processing time $T(R,S,B)$ in the incremental approach. As stated in chapter 7, regression models do not reflect arbitrary choices of B. Regarding the use of the theoretical model as an empirical model, approaches can be compared in the following way for a given aggregate, accuracy requirements (relative error ϵ and success probability of achieving it $1-\delta$), retention and slide:

1. Select regression models of applicable approaches from table :
   (a) Limit to models fitting the aggregate and / or its properties.
   (b) Limit to models supporting ϵ of the given accuracy requirements.
   (c) Limit to models supporting δ of the given accuracy requirements.
2. Estimate processing time and memory space usage from the models of the applicable approaches by the given retention and slide.
3. Determine from the plot and external constraints which approach performs best.

Clearly not all approaches apply apply. If regression models for approaches to the given aggregate exist, they can be selected. Otherwise the empirical part of the research can be repeated to obtain regression models. However, as argued for in chapter 3 aggregates with the properties of being bounded and differential will have similar performance. Hence a model for an approach to a bounded aggregate like variance is a representative for other bounded differential aggregates like average, count and sum. Note that this need not hold for other kinds of aggregates. The selection is further reduced by the accuracy requirements. The relative error bound (ϵ) states the maximum relative error by the approach.

In this research the instantaneous approach is exact ($\epsilon = 0$) and the incremental approach is exact (e.g. panes) or approximate ($\epsilon > 0$, e.g. variance histograms). By definition the relative error is an upper bound on the error, hence the approximate approaches contain the exact approaches as special cases. The success probability ($1 - \delta$) states the probability of achieving the relative error by the approach. In this research the instantaneous approach is deterministic ($1 - \delta = 1$) and the incremental approach is deterministic (e.g. panes) or randomized ($1 - \delta < 1$, e.g. sampling). By definition the success probability is a lower bound on success in achieving the error bound, hence the randomized approaches contain the deterministic approaches as special cases.

The final selection of approaches, for the given aggregate and accuracy requirements, contains regression models for processing time and memory space usage models. By using the given retention and slide, a processing time and memory space usage estimate per applicable approach is obtained. Plotting the estimates for each approach on a space-time plot, so each approach is a point on the plot representing the pair of memory space usage and processing time estimates, allows for a graphical comparison. An imaginary horizontal and vertical line through some reference point partitions the plot into four quadrants:

- Upper right: contains all approaches whose processing time and memory space usage are worse than the reference point.
- Upper left: contains all approaches whose processing time is better but memory space usage is worse than the reference point.
- Lower right: contains all approaches whose processing time is worse but memory space usage is better than the reference point.
- Lower left: contains all approaches whose processing time and memory space usage are better than the reference point.

Clearly, the approach of the reference point is "better" than the approaches of the points in the upper right quadrant. Similarly the approaches of the points in the lower left quadrant are "better" than the approach of the reference point. The approaches of the points in the upper left quadrant trade memory space usage for the benefit of processing time; a trade-off. Similarly the approaches of the points in the lower right quadrant trade processing time for the benefit of memory space usage; a trade-off. Such trade-offs require external constraints to make a final decision. The procedure for the empirical model is illustrated with an example.

Example 8.1.2. Suppose the hospital with MRI scanners from the introduction 1.1. One wants to compute the exact variance, in a deterministic fashion, over the last hour of temperature measurements sliding every 10 minutes. It is known that variance is a bounded and differential aggregate (see chapter 3). Exact variance means accuracy of 100% and thus a relative error of $\epsilon = 0$. The deterministic fashion translates into a success probability of $1 - \delta = 1$, thus $\delta = 0$. Variance over the last hour means a retention R of 60 minutes. Sliding every minute means a slide S of 10 minutes.

By the procedure the following approaches and models of table 8.1 apply:

- Time:
  - Instantaneous: $0.002 + R * 0.008 + S * 0.008$
  - Incremental with expiration list: $-0.007 + S * 0.025$
  - Incremental with panes: $0.000 + S * 0.016$
- Space:
  - Instantaneous: $26.674 + R * 1.201 + S * 0.070$
  - Incremental with expiration list: $32.389 + R * 1.354 + S * 1.426$
  - Incremental with panes: $25.842 + R * 0.002 + S * 1.201$

Knowing the models of the applicable, one can estimate the processing time and memory space usage of each approach given the retention R and slide S. Time and space estimates for each approach are plotted as depicted in 8.1. In this case the instantaneous approach has a space benefit, whereas the incremental approach with an expiration list has a time benefit with respect to one another. Hence a trade-off can be made between them. However, the incremental approach with panes has both a time and space benefit with respect to the others, hence it is the best approach to compute the aggregate variance in the given circumstances. $\diamond$

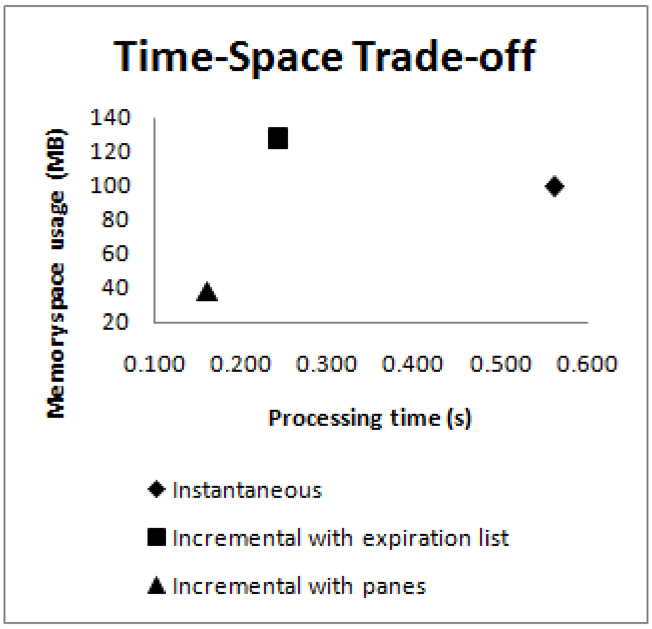


Figure 8.1: Example of space-time plot of the estimates from the models. This can be used to make trade-offs (graphically). ◦

Arguably, if two points are sufficiently close together the decision is non-obvious. Each model has an inherent error (e.g. generalization error from validation of the models), affecting the accuracy of the estimates. In a sense an estimate lies in a range of possible processing times and memory space usages as a consequence of such errors. Ideally the ranges of different approaches are disjoint and as such the error can be ignored. Especially if points are close together, a more complex comparison is required. A simple solution is to use a threshold (provided externally) so that one approach is considered better than another if their (relative) difference exceeds the threshold. Other constraints or preferences can play a role too. Clearly a simple implementation or generally applicable synopsis structure might be preferred over an aggregate-tailored synopsis structure when performance is near similar.

8.2 Generalization and Limitations

A limitation of the scalability model of the design space is that it is hard to differentiate between the performance of approaches whose computational complexities are in the same complexity class. Clearly the panes synopsis structure (see section 4.4) has update time $T_u(R) \in O(1)$ and query / lookup time $T_q(R) \in O(1)$ for computing variance. The expiration list synopsis structure (see 4.3) has similar update and query / lookup time. Nevertheless from the results in section 6.2.1 the processing time is quite different as the slide increases for a given retention, showing that panes require less processing time. A related limitation is the effect of the relative error ϵ. Variance histograms (see section 4.6) has $O(\frac{1}{\epsilon^2} \log R)$ buckets. Suppose the accuracy requirement is $\epsilon = 0.0001$, hence an accuracy of 99.99%, the number of buckets skyrockets due to the square. Clearly, on sufficiently small retentions and sufficiently large slide, the benefit of variance histograms is lost as clear from the results in section 6.2.1. However, from a scalability point of view the instantaneous approach bounds the incremental approach using variance histograms from a above. Another limitation of worst-case complexities reported in literature is that on average the costs can be (much) less. Moreover complexity is asymptotic, leaving less important terms and multiplicative constants out. Finally on a streaming data processing system, the actual implementations might show (unexpected) side-effects regarding performance.

The greatest strength of the scalability model is the independence from the environment as computational complexity deals with properties inherent to the algorithm. Moreover approaches are compared mathematically. The scalability model can be used to explain why certain performance is observed empirically or predict how performance is affected as the retention, slide and batch / small window sizes increase.

A foremost limitation of the empirical model of the design space is the limited number of regression models to approximate performance empirically. Clearly, more aggregates can be modeled and more

Processing Time, instantaneous $T(R)$ and incremental $T(R, S, B)$

Aggregate			ϵ	δ	Approach	Complexity	Regression Model (R, S in minutes)
Variance	B	D	0	0	Instantaneous	$O(R)$	0.002 + R * 0.008 + S * 0.008
Variance	B	D	0	0	Incr. / expiration list	$S * O(1) + O(1)$	-0.007 + S * 0.025
Variance	B	D	0	0	Incr. / panes	$S * O(1) + O(1)$	0.000 + S * 0.016
Variance	B	D	0.05	0.0001	Incr. / sampling	$S * O(1) + O(1)$	-0.001 + S * 0.016
Variance	B	D	0.01	0.0001	Incr. / sampling	$S * O(1) + O(1)$	0.000 + S * 0.017
Variance	B	D	0.05	0	Incr. / histogram	$S * O_{amortized}(1) + O(1)$	-0.006 + S * 0.037
Variance	B	D	0.01	0	Incr. / histogram	$S * O_{amortized}(1) + O(1)$	0.000 + R * 0.002 + S * 0.039
Quantiles	U	N	0	0	Instantaneous	$O(R * \log R + R)$	-0.145 + R * 0.049
Quantiles	U	N	0	0	Incr. / expiration list	$S * O(\log R) + O(R)$	-0.117 + 0.005 * R + 0.203 * S
Quantiles	U	N	0.05	0.0001	Incr. / sampling	$S * O(\log \lvert Sample \rvert) + O(\lvert Sample \rvert)$	0.002 + 0.051 * S
Quantiles	U	N	0.01	0.0001	Incr. / sampling	$S * O(\log \lvert Sample \rvert) + O(\lvert Sample \rvert)$	0.033 + 0.113 * S
Quantiles	U	N	0.05	0	Incr. / quantile summary	N/A	0.003 + R * 0.001 + S * 0.137
Quantiles	U	N	0.01	0	Incr. / quantile summary	N/A	-0.024 + S * 0.198

Memory Space Usage, instantaneous $S(R)$ and incremental $S(R, B)$

Aggregate			ϵ	δ	Approach	Complexity	Regression Model (R, S in minutes)
Variance	B	D	0	0	Instantaneous	$O(R)$	26.674 + R * 1.201 + S * 0.070
Variance	B	D	0	0	Incr. / expiration list	$O(R)$	32.389 + R * 1.354 + S * 1.426
Variance	B	D	0	0	Incr. / panes	$O(\frac{R}{B})$	25.842 + R * 0.002 + S * 1.201
Variance	B	D	0.05	0.0001	Incr. / sampling	$O(\lvert Sample \rvert)$	25.899 + S * 1.207
Variance	B	D	0.01	0.0001	Incr. / sampling	$O(\lvert Sample \rvert)$	27.003 + R * -0.005 + S * 1.306
Variance	B	D	0.05	0	Incr. / histogram	$O(\epsilon^{-2} * \log R)$	28.810 + R * 0.055 + S * 2.846
Variance	B	D	0.01	0	Incr. / histogram	$O(\epsilon^{-2} * \log R)$	34.253 + R * 1.137 + S * 1.271
Quantiles	U	N	0	0	Instantaneous	$O(R)$	27.544 + R * 2.122
Quantiles	U	N	0	0	Incr. / expiration list	$O(R)$	65.269 + R * 3.213
Quantiles	U	N	0.05	0.0001	Incr. / sampling	$O(\lvert Sample \rvert)$	30.432 - 0.063 * R + 1.276 * S
Quantiles	U	N	0.01	0.0001	Incr. / sampling	$O(\lvert Sample \rvert)$	74.234 - 0.082 * R + 0.614 * S
Quantiles	U	N	0.05	0	Incr. / quantile summary	$O(\epsilon^{-1} * \log \epsilon^{-1} * \log R)$	27.842 + R * 0.027 + S * 1.138
Quantiles	U	N	0.01	0	Incr. / quantile summary	$O(\epsilon^{-1} * \log \epsilon^{-1} * \log R)$	32.389 + S * 1.221

Table 8.1: Overview of complexity (scalabilty) and regression models (empirical approximations) for the bounded differential aggregate variance and unbounded non-differential aggregate quantiles. The complexities come from chapters 3a and 4. The regression models come from chapter 7 and are based on empirical results in chapter 6. Processing time complexity and regression models should be interpreted as $T(R)$ for the instantaneous approach and $T(R, S, B)$ for the incremental approach. Memory space usage complexity and regression models should be interpreted as $S(R)$ for the instantaneous approach and $S(R, B)$ for the incremental approach. Let B denote bounded, U unbounded, D differential and N non-differential for the aggregates. Let ϵ denote the relative error upper bound and $1 - \delta$ the success probability lower bound of achieving it. Let $\lvert Sample \rvert$ be in $O(\log(\epsilon^{-2} * \log(\epsilon * \delta)^{-1}))$. Note that in the regression models R denotes the retention in minutes and S is the slide in minutes at a rate of about 320 elements per second. Note that in chapters 5, 6 and 7 it is argued that batch size B is equal to slide S during measurements, which most likely affect memory space usage of the incremental approach. It seems insignificant to processing time. As a consequence regression models do not reflect arbitrary choices of batch size. ∘

synopsis structures can be used to further differentiate the incremental approach. A related limitation stems from the measurements on which the models are based. Ideally more workloads, i.e. more variety in retention and slide, can be measured and used in the derivation of the models to better cover performance under different circumstances. The current measurements allow only for limited interpolation and extrapolation. The statistical limitations from models, i.e. heteroscedasticity, autocorrelation and non-normality (see chapter 7), can perhaps be mitigated by such better coverage of the measurements. On the other hand, a more sophisticated analysis might be worthwhile. Perhaps an analysis less constrained by assumptions of regression analysis can be tried or a more sophisticated variant of regression analysis. Another clear limitation of the models is their empirical nature. The measurements depend on particular implementations and a particular environment (streaming data processing system, operating system, hardware). That is why the scalability and empirical model instantiations of the theoretical model are complements of each other.

Nevertheless, the regression models can be generalized to some extent. In particular the models for variance can be generalized. The sum, count, average, skewness, kurtosis and other bounded differential aggregates share some important traits: constant size aggregates and incremental/decremental formula to update them (see chapter 3). Hence they will have similar performance on expiration lists, panes and the instantaneous approach. Perhaps the variance histograms can be altered or extended to support such aggregates. This generalization is more difficult for other kinds of aggregates. In a limited way the environment can be generalized. Suppose each operation is performed k times faster, e.g. due to hardware improvements. This effectively scales the measurements linearly. If only some operations become faster or the streaming data processing system radically changes the way data is processed, the models lose their power.

A key insight from the use of the theoretical model is that the instantaneous approach and incremental approach both have their uses depending on the circumstances, e.g. algorithms used, output semantics. Even though the instantaneous approach seems naive, results alone (see chapter 6) show it can outperform the incremental approach if the slide is large enough and vice versa.

8.3 Guidelines

Throughout the chapters, various guidelines were derived. The guidelines augment the design space. They can be used in the decision making process of which approach performs best. The guidelines can be interpreted as lessons learned or rules of thumb. Some interesting guidelines are highlighted below, the respective chapters provide additional guidelines:

- The instantaneous approach requires processing time and memory space usage in $\Omega(R)$ (see chapter 3).
- Exact computation with a slide of 1 element has memory space usage $\Omega(R)$, justifying approximations trading accuracy for reduced memory space usage (see chapter 3).
- Synopsis structures must support a notion of retention and implicit deletes, hence sliding windows, to be applicable under the incremental approach (see chapter 4).
- As the slide (and batch) gets sufficiently large with respect to the retention, the instantaneous approach outperforms the incremental approach regarding processing time and / or memory space usage (see chapter 6). This can be assessed by the theoretical model and table 8.1 showing the computational complexity and regression models of performance in practice regarding the approaches.

Chapter 9

Conclusions

This chapter will answer the main research question after answering each sub research question in section 9.1. It ends with suggestions for future work in section 9.2.

9.1 Answers to Research Questions

The discussion is started with answers to the sub questions and finally answer the main question.

How is the computation of aggregates related to the approaches?

The aggregates are computed over a retention containing R elements of interest that slides as S new elements become available. Conceptually the retention behaves like a sliding window and the approaches determine the way in which this is materialized:

- Instantaneous approach: a large window covering the retention is kept in memory over which an aggregate is computed independent of previous computations.
- Incremental approach: a synopsis structure covering the retention is kept in memory which is updated by batches / small windows (accounting for new and expired data) of B elements, where $1 \leq B \leq \gcd(R, S)$.

The semantics were defined formally in 2. An abstract theoretical model is derived in chapter 3, relating the approaches to the computation of aggregates in terms of performance costs. For the instantaneous approach the average processing time is $T(R)$: the time to compute the aggregate over R elements. The memory space usage is $S(R)$: the space to store the large window of R elements and the algorithm to compute the aggregate over R elements. For the incremental approach the average processing time is $T(R, S, B) = S * T_u(R) + T_q(R)$: for each S elements in the slide the time to update the synopsis structure over R elements ($T_u(R)$) plus the time for a query / lookup of the aggregate from the synopsis structure over R elements ($T_q(R)$). The memory space usage is $S(B, R)$: the space to store a batch / small window of B elements and the synopsis structure over R elements.

The theoretical model can be used to compare the approaches in terms of performance in practice by replacing the generic time $T(\cdot)$ and space $S(\cdot)$ functions in the model with empirical approximations. On the other hand the theoretical model can be used to compare the approaches in scalability by replacing generic time $T(\cdot)$ and space $S(\cdot)$ functions in the model with computational complexity. It is clear that the instantaneous approach takes processing time and memory space usage in $\Omega(R)$. In the incremental approach as S goes to 1 in $T(R, S, B)$, the overlap between consecutive retentions is minimized. Exact computations for $S = 1$ require space $\Omega(R)$ mentioned in literature. This justifies approximations trading accuracy for reduced memory space usage. As S goes to R in $T(R, S, B)$ the overlap between consecutive retentions is maximized and processing time becomes $\Omega(R)$. Small batches are justified to reduce memory space usage.

Two properties of aggregates mentioned in literature are of main interest regarding exact computation: boundedness in definition 3.4.1 and differentiality in definition 3.4.4. Boundedness does not apply to the instantaneous approach. In the incremental approach the batches perform a partitioning of the retention. It is shown that regardless of the boundedness memory space usage is in $\Omega(R)$. Both approaches can take advantage of differentiality. The instantaneous approach can use the incremental formula to compute an aggregate in one pass over the retention, if a constant time function exists (e.g. for variance). The incremental approach can use the property to apply the effects of new elements to and undo the effects of expired elements from an internally maintained aggregate. A differential aggregate has $T_q(R) \in O(1)$ to retrieve an internally maintained aggregate and a bounded differential aggregate has $T_u(R) \in O(1)$ by constant time incremental and decremental functions. A non-differential aggregate has $T_u(R) \in \Omega(1)$ and $T_q(R) \in \Omega(R)$. Concrete algorithms for aggregates can be used to tighten bounds.

From the aggregates of interest to Info Support, variance is a bounded differential aggregate and quantiles an unbounded non-differential aggregate. Variance is computed similar to average, count, sum also of interest to Info Support. Quantiles cover median, minimum and maximum as particular cases also of interest to Info Support. Those aggregates are used to guide the discussion on synopsis structures and for the empirical part of the research.

How are synopsis structures related to the incremental approach?

Synopsis structures are discussed in chapter 4. Under the incremental approach the synopsis structure must cover the retention. To this end new elements are presented explicitly, by batches. Old elements expire implicitly, i.e. the synopsis structure must know which elements expire and how to handle them. In a count-based retention the number of elements is fixed, while in a count-based distinct-time and time-based retention the number of elements is variable. Synopsis structures supporting the latter, support the former as a special case. The opposite does not necessarily hold, though might with a few adaptations. Effectively synopsis structures must support sliding windows. Another important issue in synopsis structures is the relative error bound (ϵ) and the success probability of achieving the error bound ($1 - \delta$) of aggregates determined from them, i.e. how well they reflect the retention. The following taxonomy is proposed:

- Fixed number of elements (in the retention):
  - Exact ($\epsilon = 0$): expiration list (see section 4.3) and panes (see section 4.4).
  - Approximate ($\epsilon > 0$)
    - Deterministic ($1 - \delta = 1$): variance histograms (see section 4.6) and quantile summaries (see section 4.7).
    - Randomized or probablistic ($1 - \delta < 1$): sampling (see section 4.5).
- Variable number of elements (in the retention):
  - Idem.

A wealth of synopsis structures is present in literature. In this research the discussion is confined to synopsis structures in the context of the aggregates variance and quantiles. Moreover synopsis structures covering the taxonomy are chosen to provide Info Support a broad view on the matter, rather than treating a particular synopsis structure and its optimizations in depth.

What is the performance of the approaches in practice (empirically)?

Performance in practice, hence empirically, allows one to differentiate the performance of approaches belonging to the same computational complexity class, observe more average-case like performance and observe (unexpected) side-effects of performance on a system. A practical way to assess performance of the approaches is by measuring actual processing time of computing the aggregate over the retention and measuring memory space usage in the system as explained in chapter 5. For the selected aggregates variance and quantiles this is performed on the StreamInsight 1.1 streaming data processing system fitting into the Business Intelligence stack of Info Support. As user defined aggregates in StreamInsight require output for every window, the incremental approach might output more frequently than the instantaneous approach if multiple batches / small windows cover a slide. This would break output semantics. As a consequence the batch sizes were equal to the slide. Looking at the theoretical model, this most likely affect memory space usage $S(B, R)$ in the incremental approach. It depends directly on the batch size and synopsis structure covering the retention. For processing time $T(R, S, B) = S * T_u(R) + T_q(R)$, of the incremental approach, the batch size seems insignificant. Larger slides and smaller batch sizes cannot be observed in the proposed setup of chapter 5. The results are in chapter 6. Some of the most important observations from the results are:

- The instantaneous approach has processing time and memory space usage grow (linearly) with the retention and is independent of the slide. This confirms the theoretical model for the instantaneous approach $T(R)$ and $S(R)$, which is only dependent on the retention. It is in line with proposition 3.2.4.

- As the slide (and batch) increases with respect to the retention, the processing time to update the synopsis structure and memory space usage increase under the incremental approach. The empirical results confirm the theoretical model for the incremental approach. As $T(R, S, B) = S * T_u(R) + T_q(R)$, the effect of a growing S is clear. Memory space usage $S(B, R)$ grows as batches get larger.

- The use of (biased reservoir) sampling as a synopsis structure provides the best processing time and memory space usage while achieving high accuracy (though probablistically) for both quantiles and variance:

  - For variance, processing time of sampling lost its benefit at about $B = S \approx \frac{1}{2} * R$, while having clear memory space usage benefits up to the same point. This should be representative for other bounded differential aggregates.
  - For quantiles, processing time of sampling lost its benefit at about $B = S \approx \frac{2}{5} * R$ at high levels of accuracy, while having clear memory space usage benefits up to $B = S \approx \frac{1}{2} * R$.

- The use of panes as a synopsis structure provides the best processing time and memory space usage for exact computation of variance when the batches are sufficiently large. This should be representative for other bounded differential aggregates. Its performance was similar to sampling.

- The use of expiration lists as a (trivial) synopsis structure has processing time lost its benefit at about $B = S \approx \frac{1}{3} * R$ for variance, representative for other bounded aggregates, and at about $B = S \approx \frac{1}{5} * R$ for quantiles. There are no memory space usage benefits as all elements in the retention are stored, similar to a large window of the instantaneous approach.

- The use of variance histograms as a synopsis structure lost its processing time benefits at about $B = S \approx \frac{1}{8} * R$ at high levels of accuracy and lost memory space usage benefits at about $B = S \approx \frac{2}{5} * R$ on lower levels of accuracy. It seems the memory space benefits at higher levels of accuracy require very large retentions. Accuracy is deterministic and very good.

- The use of quantile summaries as a synopsis structure lost its processing time benefits at about $B = S \approx \frac{1}{4} * R$ at high levels of accuracy, while having clear memory space usage benefits up to $B = S \approx \frac{1}{2} * R$ (similar to sampling). Accuracy is deterministic and very good.

Note the results are tied to the particular implementations and environment. Most importantly, the results show both the instantaneous and incremental approaches have processing time and / or memory space benefits mainly dependent on the size of the batches and slide (or inversely: overlap). To generalize the measurements on the environment, regression models are derived as discussed in chapter 7: a processing time and memory space usage model per aggregate per accuracy requirements and per approach. The regression models provide empirical approximations of the performance of an aggregate that can be plugged into the theoretical model to obtain an empirical model. The processing time model approximates $T(R)$ and $T(R, S, B)$ for the instantaneous and incremental approaches respectively, whereas the memory space usage model approximates $S(R)$ and $S(B, R)$. The models do not reflect arbitrary choices of batch size B, due to the setup of the measurements proposed in chapter 5 and hence results from which the models are derived. The assumptions of homoscedasticity, independence and normality of the residuals (discussed in chapter 7) were often violated and sometimes a nonlinear relationship was present. As a consequence the regression models have limited statistical strength. The estimators for the coefficients in the model might not be the best linear unbiased estimators (BLUE). The models are most suited to local approximations (local to the range of measurements). More extensive empirical evaluation with larger variety in retention and slide is recommended. Additionally more advanced analysis could be employed like generalized least squares (GLS), nonlinear and nonparametric regression that provide robustness to (some of the) assumptions. Validation by an independent set of measurements shows an error similar to the (standard) error (of the estimate) for the model from regression analysis. This indicates the models perform conform expectation on unseen data.

What is a design space for computing aggregates under the approaches?

The design space at an abstract level is the use of the theoretical model of chapter 3 to compare the approaches and determine which one has the best performance, regarding processing time and memory space usage. Moreover it contains the guidelines derived during the research. An instantiation of the theoretical model with computational complexity for $T(\cdot)$ and $S(\cdot)$ of the approaches yields the scalability model. This can be used to compare approaches in terms of $f \in \Omega(g), f \in \Theta(g)$ and $f \in O(g)$; how the approaches scale with respect to one another in a mathematical way independent of the environment (streaming data processing system, implementation). Computational complexity of the approaches is summarized in table 8.1 and chapters 3 and 4.

An instantiation of the theoretical model with the empirical approximations, i.e. the regression models, allows performance estimates to be compared. Given an aggregate and accuracy requirements (relative error bound ϵ and success probability of achieving the relative error bound $1 - \delta$), the regression models of applicable approaches can be selected. By a given retention and slide, a processing time and memory space usage estimate per approach is obtained. The estimates can be used to determine which approach performs better than another by comparing the estimates. Trade-offs of processing time and memory space usage must be resolved by external constraints, e.g. from a business case stating preferences. Regression models for empirical approximations are summarized in table 8.1 and chapters 6 and 7.

The guidelines augment the design space and can be interpreted as lessons learned of rules of thumb. Some interesting guidelines are:

- The instantaneous approach requires processing time and memory space usage in $\Omega(R)$ (see chapter 3).
- Exact computation with a slide of 1 element has memory space usage $\Omega(R)$, justifying approximations trading accuracy for reduced memory space usage (see chapter 3).
- Synopsis structures must support a notion of retention and implicit deletes, hence sliding windows, to be applicable under the incremental approach (see chapter 4).
- As the slide (and batch) gets sufficiently large with respect to the retention, the instantaneous approach outperforms the incremental approach regarding processing time and / or memory space usage (see chapter 6). This can be assessed by the theoretical model and table 8.1 summarizing computational complexity and regression models of performance in practice regarding the approaches.

Main question: How to compute an aggregate over sliding windows?

In the context of streaming data processing with limited resources, performance is considered important. Consequently the main question is answered from the perspective of the performance, i.e. the best performing approach to compute an aggregate. Two conceptually approaches were identified: the instantaneous and incremental approach. Their output semantics are the same, however their performance and implementation is not.

The design space is proposed to determine which approach performs best and answers how to compute an aggregate over sliding windows by means of performance comparison. The use of the theoretical model, either instantiated as the scalability or empirical model, is used to assess and compare the performance of the approaches. A key insight confirmed by the empirical results is that both approaches have their uses depending on the circumstances. In general as the slide S and batch B increase with respect to the retention R, the instantaneous approach has performance closing in or improving upon the incremental approach and vice versa.

9.2 Future Work

In this research performance of the approaches in isolation was assessed. However, in a practical setting a system can work on many aggregates or other kinds of computations simultaneously. Reducing processing time and memory space usage can scale up the number of simultaneous aggregates being computed. The extent to which this can be achieved, cannot be answered by the results of this research. Moreover, sharing resources like synopsis structures across different computations is not investigated, e.g. sharing a sample over the retention to compute aggregates simultaneously.

Regarding empirical results other distributions of data (e.g. Gaussian or non-synthetic), other aggregates and synopsis structures, more accuracy levels (99.99%) that may shift performance in the favor of the instantaneous approach, different success probabilities ($\delta \neq 0.0001$) and more combinations of retention and slide can increase the insight into the performance of the approaches. The extreme cases can be explored, the performance of the approaches when sliding 1 element at a time (issues discussed in chapter 3). Many more aggregates and synopsis structures exist. They can be compared by the theoretical model and be accounted for in the design space. However analogous to variance representing bounded differential aggregates, it would be interesting to have more classes of aggregates to abstract from individual aggregates. Moreover other synopsis structures might be worthwhile to investigate, like wavelets [45]. The synopsis structures like the quantile summaries, are based on a technique of intervals [5, 7] that is more broadly applicable than to just quantiles. Moreover it can be used to approximate aggregates over the last $n \leq N$ elements, called the $n - of - N$ model [49]. Potentially this can generalize synopsis structures or techniques on which they are based. Other issues are parallelization and composability of synopsis structures, i.e. multithreading and combining synopsis structures over disjoint parts of data.

A drawback of the empirical results by measuring actual processing time using a high resolution timer and memory space usage using information from the operating system is the tight bond with the environment and implementations. Measuring execution counts of (parts) of the implementation abstracts from the environment and is not susceptible to issues of resolution (like fluctuations in actual time or memory space usage) [37].

The regression analysis in this research is hindered by the assumptions of homoscedasticity (constant variance), independence (autocorrelation) and normality of the residuals. This affects the statistical strength of the regression models and limits the ability to obtain the best coefficients. Perhaps other statistical methods than regression analysis can be explored or more specific forms of regression analysis can be employed that are robust to violations of at least some of the assumptions. Another drawback of the regression models is their limited applicability. In a sense they provide local approximations, local to the range of measurements. This affects the quality of interpolation and extrapolation by the regression models. For practical purposes, in particular regarding Info Support, the regression models should be tailored to particular business cases. This requires the empirical part to be repeated.

References

[1] Charu C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 607–618. VLDB Endowment, 2006.

[2] Charu C. Aggarwal and Philip S. Yu. A survey of synopsis construction in data streams. In *Data Streams: Models and Algorithms*, chapter 9.

[3] M. H. Ali, C. Gerea, B. S. Raman, B. Sezgin, T. Tarnavski, T. Verona, P. Wang, P. Zabback, A. Ananthanarayan, A. Kirilov, M. Lu, A. Raizman, R. Krishnan, R. Schindlauer, T. Grabs, S. Bjeletich, B. Chandramouli, J. Goldstein, S. Bhat, Ying Li, V. Di Nicola, X. Wang, David Maier, S. Grell, O. Nano, and I. Santos. Microsoft cep server and online behavioral targeting. *Proc. VLDB Endow.*, 2(2):1558–1561, 2009.

[4] Mohamed H. Ali, Badrish Chandramouli, Balan Sethu Raman, and Ed Katibah. Spatio-temporal stream processing in microsoft streaminsight. *IEEE Data Eng. Bull.*, 33(2):69–74, 2010.

[5] Arvind Arasu and Gurmeet Singh Manku. Approximate counts and quantiles over sliding windows. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '04, pages 286–296, New York, NY, USA, 2004. ACM.

[6] Arvind Arasu and Jennifer Widom. A denotational semantics for continuous queries over streams and relations. *SIGMOD Rec.*, 33(3):6–11, 2004.

[7] Arvind Arasu and Jennifer Widom. Resource sharing in continuous sliding-window aggregates. In *VLDB*, pages 336–347, 2004.

[8] Sara Baase and Allen Van Gelder. *Computer Algorithms: Introduction to Design and Analysis.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1999.

[9] Brain Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '03, pages 234–243, New York, NY, USA, 2003. ACM.

[10] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, New York, NY, USA, 2002. ACM.

[11] Brian Babcock, Mayur Datar, and Rajeev Motwani. Sampling from a moving window over streaming data. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '02, pages 633–634, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

[12] J. Bennett, R. Grout, P. Pebay, D. Roe, and D. Thompson. Numerically stable, single-pass, parallel statistics algorithms. In *Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on*, 31 2009.

[13] W.D. Berry and S. Feldman. Multiple regression in practice. volume 07 of *Sage University Paper Series on Quantitative Applications in Social Sciences*, 1985.

[14] Irina Botan, Roozbeh Derakhshan, Nihal Dindar, Laura M. Haas, Renée J. Miller, and Nesime Tatbul. Secret: A model for analysis of the execution semantics of stream processing systems. *PVLDB*, 3(1):232–243, 2010.

[15] Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '09, pages 147–156, New York, NY, USA, 2009. ACM.

[16] Paul G. Brown and Peter J. Haas. Techniques for warehousing of sample data. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 6–, Washington, DC, USA, 2006. IEEE Computer Society.

[17] Chiranjeeb Buragohain and Subhash Suri. Quantiles on streams. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 2235–2240. Springer US, 2009.

[18] Don Carney, Uğur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Monitoring streams: a new class of data management applications. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 215–226. VLDB Endowment, 2002.

[19] Sharma Chakravarthy and Qingchun Jiang. *Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing.* Springer Publishing Company, Incorporated, 2009.

[20] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. Technical report, Stanford, CA, USA, 1979.

[21] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel R. Madden, Fred Reiss, and Mehul A. Shah. Telegraphcq: continuous dataflow processing. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 668–668, New York, NY, USA, 2003. ACM.

[22] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. Niagaracq: a scalable continuous query system for internet databases. *SIGMOD Rec.*, 29:379–390, May 2000.

[23] Edith Cohen and Martin J. Strauss. Maintaining time-decaying stream aggregates. *J. Algorithms*, 59:19–36, April 2006.

[24] Patricia Cohen, Jacob Cohen, Stephen G. West, and Leona S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, Third Edition.* Lawrence Erlbaum, third edition, 2002.

[25] Sara Cohen. User-defined aggregate functions: bridging theory and practice. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 49–60, New York, NY, USA, 2006. ACM.

[26] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '06, pages 263–272, New York, NY, USA, 2006. ACM.

[27] Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31:1794–1813, June 2002.

[28] Mayur Datar and Rajeev Motwani. The sliding-window computation model and results. In Charu C. Aggarwal, editor, *Data Streams*, volume 31 of *Advances in Database Systems*, pages 149–167. Springer US, 2007.

[29] Rainer Gemulla and Wolfgang Lehner. Sampling time-based sliding windows in bounded space. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 379–392, New York, NY, USA, 2008. ACM.

[30] Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. A dip in the reservoir: maintaining sample synopses of evolving datasets. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 595–606. VLDB Endowment, 2006.

[31] Rainer Gemulla, Wolfgang Lehner, and Peter J. Haas. Maintaining bernoulli samples over evolving multisets. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '07, pages 93–102, New York, NY, USA, 2007. ACM.

[32] Thanaa M. Ghanem, Walid G. Aref, and Ahmed K. Elmagarmid. Exploiting predicate-window semantics over data streams. *SIGMOD Rec.*, 35:3–8, March 2006.

[33] Phillip B. Gibbons and Yossi Matias. Synopsis data structures for massive data sets. In James M. Abello and Jeffrey Scott Vitter, editors, *External memory algorithms*, pages 39–70. American Mathematical Society, Boston, MA, USA, 1999.

[34] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27:261–298, September 2002.

[35] Lukasz Golab, David DeHaan, Erik D. Demaine, Alejandro Lopez-Ortiz, and J. Ian Munro. Identifying frequent items in sliding windows over on-line packet streams. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 173–178, New York, NY, USA, 2003. ACM.

[36] Lukasz Golab and M. Tamer Özsu. Issues in data stream management. *SIGMOD Rec.*, 32(2):5–14, 2003.

[37] Simon F. Goldsmith, Alex S. Aiken, and Daniel S. Wilkerson. Measuring empirical computational complexity. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ESEC-FSE '07, pages 395–404, New York, NY, USA, 2007. ACM.

[38] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1:29–53, 1997. 10.1023/A:1009726021843.

[39] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. *SIGMOD Rec.*, 30:58–66, May 2001.

[40] Sudipto Guha and Nick Koudas. Approximating a data stream for querying and estimation: Algorithms and performance evaluation. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 567–, Washington, DC, USA, 2002. IEEE Computer Society.

[41] Damodar N. Gujarati. *Basic Econometrics*, chapter 11–12. Mcgraw-Hill, May 2002.

[42] Jiawei Han. *Data Mining: Concepts and Techniques, Second Edition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[43] Andrew C. Harvey. *The Econometric Analysis of Time Series, 2nd Edition*, volume 1. The MIT Press, 2 edition, 1990.

[44] Yannis Ioannidis. The history of histograms (abridged). In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 19–30. VLDB Endowment, 2003.

[45] Ying-Hui Kong, Jin-Sha Yuan, Lei Wu, and Tie-Feng Zhang. A method for continuous query over data stream using wavelet synopsis. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 7, pages 4119 –4123, aug. 2007.

[46] Jürgen Krämer and Bernhard Seeger. Semantics and implementation of continuous sliding window queries over data streams. *ACM Trans. Database Syst.*, 34(1):1–49, 2009.

[47] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A. Tucker. No pane, no gain: efficient evaluation of sliding-window aggregates over data streams. *SIGMOD Record*, 34:2005, 2005.

[48] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A. Tucker. Semantics and evaluation techniques for window aggregates in data streams. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 311–322, New York, NY, USA, 2005. ACM.

[49] Xuemin Lin, Hongjun Lu, Jian Xu, and Jeffrey Xu Yu. Continuously maintaining quantile summaries of the most recent n elements over a data stream. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 362–374, Washington, DC, USA, 2004. IEEE Computer Society.

[50] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[51] Joaquim P. Marques de Sá. *Applied Statistics Using SPSS, STATISTICA, MATLAB and R.* Springer, 1 edition, 2003.

[52] Microsoft. Developer's guide (streaminsight) - sql server 2008 r2. `http://http://msdn.microsoft.com/en-us/library/ee391564.aspx`. Accessed October 25, 2010.

[53] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Singh Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query processing, approximation, and resource management in a data stream management system. In *CIDR*, number 2002-41, 2003.

[54] J. I. Munro and M. S. Paterson. Selection and sorting with limited storage. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 253–258, Washington, DC, USA, 1978. IEEE Computer Society.

[55] NIST/SEMATECH. Engineering statistics handbook - data analysis for process modeling. `http://www.itl.nist.gov/div898/handbook/pmd/section4/pmd4.htm`. Accessed February 15, 2011.

[56] Kostas Patroumpas and Timos Sellis. Window specification over data streams. In Torsten Grust, Hagen Höpfner, Arantza Illarramendi, Stefan Jablonski, Marco Mesiti, Sascha Müller, Paula Lavinia Patranjan, Kai-Uwe Sattler, Myra Spiliopoulou, and Jef Wijsen, editors, *Current Trends in Database Technology – EDBT 2006*, volume 4254 of *Lecture Notes in Computer Science*, pages 445–464. Springer Berlin / Heidelberg, 2006.

[57] Lin Qiao, Divyakant Agrawal, and Amr El Abbadi. Supporting sliding window queries for continuous data streams. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, SSDBM '03, pages 85–96, Washington, DC, USA, 2003. IEEE Computer Society.

[58] Alex Raizman, Asvin Ananthanarayan, Anton Kirilov, Badrish Chandramouli, and Mohamed Ali. An extensible test framework for the microsoft streaminsight query processor. In *Proceedings of the Third International Workshop on Testing Database Systems*, DBTest '10, pages 2:1–2:6, New York, NY, USA, 2010. ACM.

[59] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining.* ADDISON WESLEY PUBLISHERS, 2006.

[60] Peter A. Tucker, David Maier, Tim Sheard, and Leonidas Fegaras. Exploiting punctuation semantics in continuous data streams. *IEEE Trans. on Knowl. and Data Eng.*, 15:555–568, March 2003.

[61] Princeton University. Introduction to regression. `http://dss.princeton.edu/online_help/analysis/regression_intro.htm`. Accessed January 25, 2011.

[62] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11:37–57, March 1985.

[63] John B. Willett and Judith D. Singer. Another cautionary note about r2: Its use in weighted least-squares regression analysis. *The American Statistician*, 42(3):pp. 236–238, 1988.

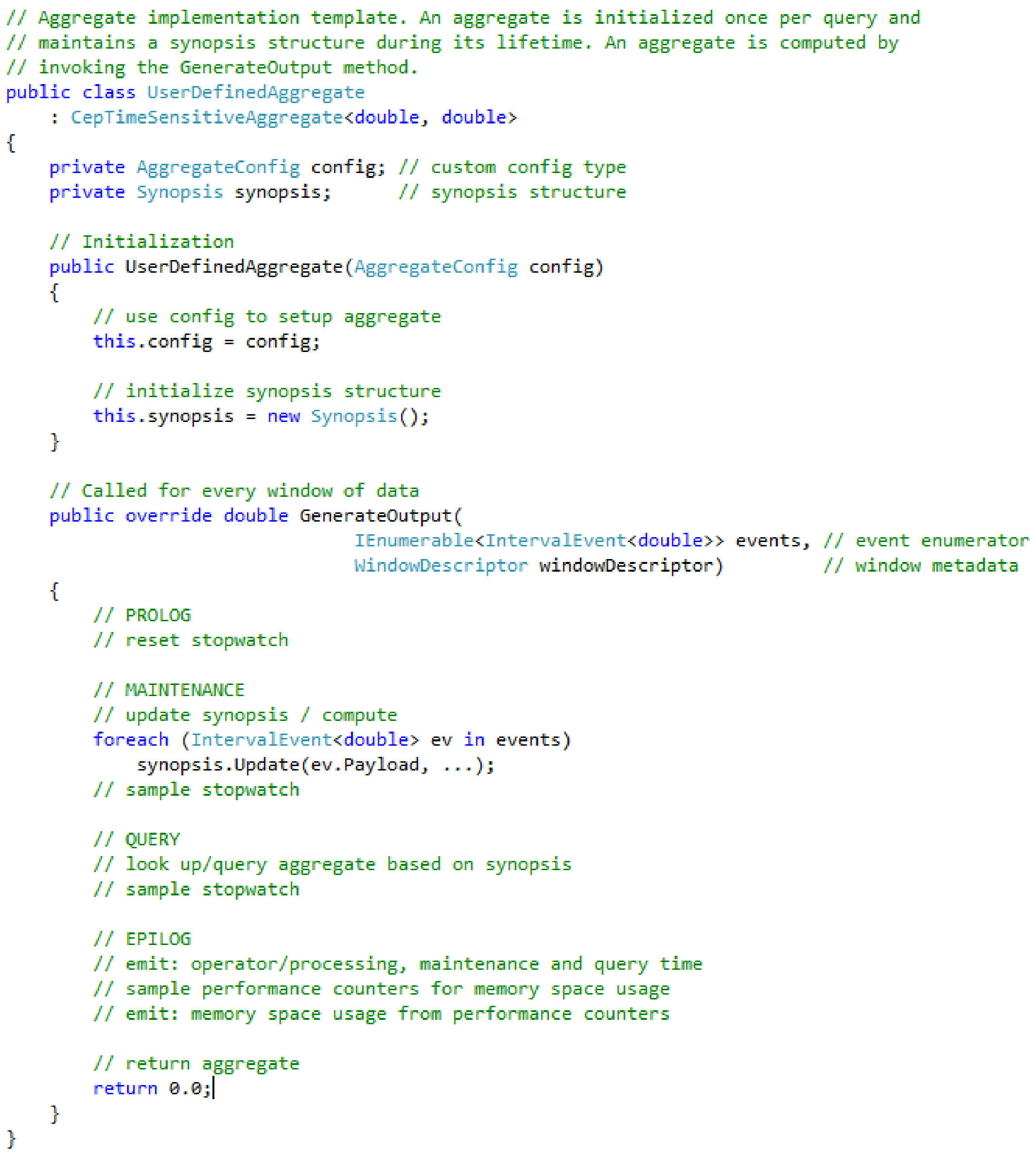
[64] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, 2005.

[65] Linfeng Zhang and Yong Guan. Variance estimation over sliding windows. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '07, pages 225–232, New York, NY, USA, 2007. ACM.

[66] Yunyue Zhu and Dennis Shasha. Statstream: statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 358–369. VLDB Endowment, 2002.

Appendix A

User Defined Aggregate Code Template



```csharp
// Aggregate implementation template. An aggregate is initialized once per query and
// maintains a synopsis structure during its lifetime. An aggregate is computed by
// invoking the GenerateOutput method.
public class UserDefinedAggregate
    : CepTimeSensitiveAggregate<double, double>
{
    private AggregateConfig config; // custom config type
    private Synopsis synopsis;      // synopsis structure

    // Initialization
    public UserDefinedAggregate(AggregateConfig config)
    {
        // use config to setup aggregate
        this.config = config;

        // initialize synopsis structure
        this.synopsis = new Synopsis();
    }

    // Called for every window of data
    public override double GenerateOutput(
                            IEnumerable<IntervalEvent<double>> events, // event enumerator
                            WindowDescriptor windowDescriptor)         // window metadata
    {
        // PROLOG
        // reset stopwatch

        // MAINTENANCE
        // update synopsis / compute
        foreach (IntervalEvent<double> ev in events)
            synopsis.Update(ev.Payload, ...);
        // sample stopwatch

        // QUERY
        // look up/query aggregate based on synopsis
        // sample stopwatch

        // EPILOG
        // emit: operator/processing, maintenance and query time
        // sample performance counters for memory space usage
        // emit: memory space usage from performance counters

        // return aggregate
        return 0.0;
    }
}
```

Figure A.1: A C# code template for a user defined aggregate (UDA) in StreamInsight v1.1 used during the research. A query has an instance of the aggregate. A configuration and synopsis is kept per aggregate instance. Every time a window is processed, the GenerateOutput method is invoked by StreamInsight. ◦

Appendix B

Regression Plots

The scedasticity plots of residuals against predicted values or the independent values are used to demonstrate the spread of the residuals. A uniform spread of the residuals indicates homoscedasticity, a non-uniform spread and/or non-constant spread indicates heteroscedasticity. Funnel patterns are typical for heteroscedasticity. The histogram with normal curve demonstrates the extent to which the residuals are normally distributed. The better the bars of the histogram follow the curve, the more closely the residuals are distributed normally and vice versa. A related plot is the normal P-P plot where the straight line represents the normal distribution. Ideally the residuals are on the straight line. The more the residuals deviate from the straight line, the less likely they are normally distributed. Lag plots show the residuals against the lagged residuals (e_i against e_{i-1}). A uniformly scattered lag plot indicates independence of the residuals, hence absence of autocorrelation and a Durbin-Watson statistic towards 2. A pattern showing points from low to high (left to right) indicates positive autocorrelation and a Durbin-Watson statistic towards 0; a pattern showing points from high to low (left to right) indicates negative autocorrelation and a Durbin-Watson statistic towards 4. Positive and negative autocorrelation indicate a lack of independence of the residuals.

B.1 Processing Time: Variance

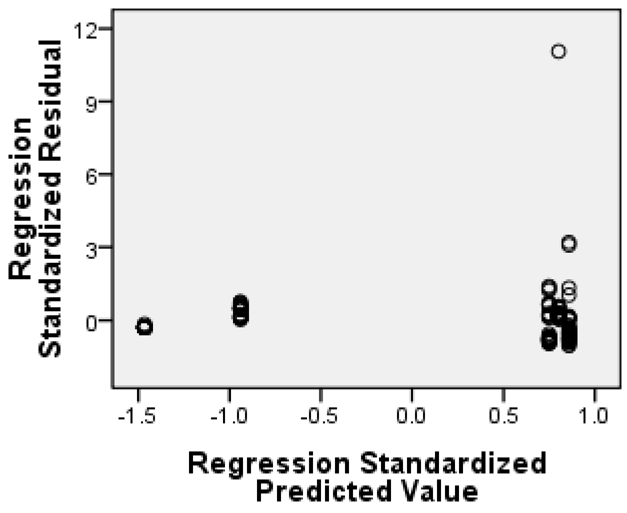


Figure B.1: Residuals against predicted values plot for the processing time model of the instantaneous approach. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

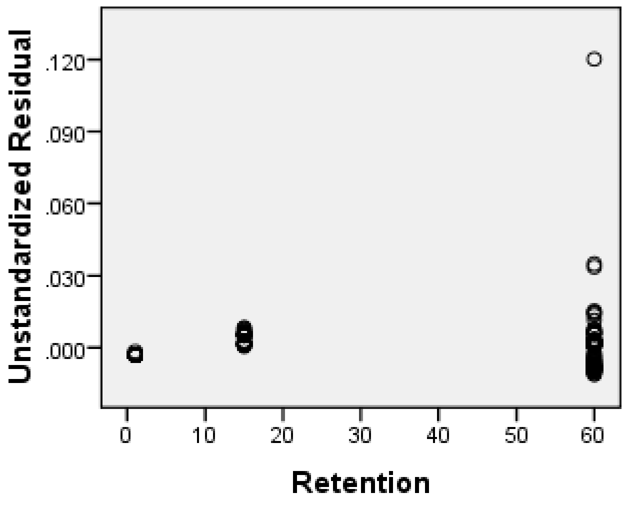


Figure B.2: Residuals-retention plot for the processing time model of the instantaneous approach. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

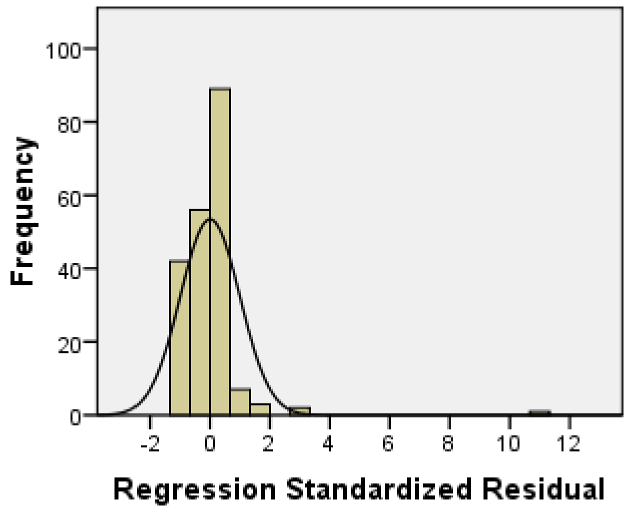


Figure B.3: Histogram of the standardized residuals with normal curve for the processing time model of the instantaneous approach. The curve is not approximated well. ◦

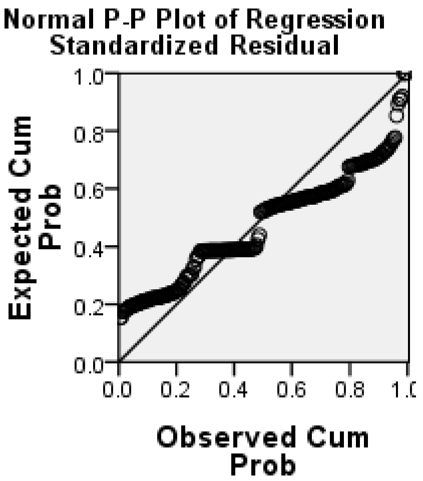


Figure B.4: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the instantaneous approach. The points are clearly off the straight line, indicating deviation from the normal distribution. ◦

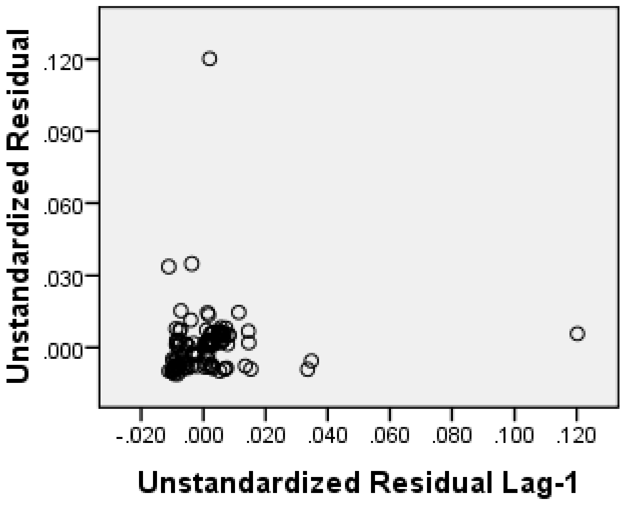


Figure B.5: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the instantaneous approach. The plot indicates hardly any autocorrelation, which is in line with the Durbin-Watson statistic towards 2. ◦

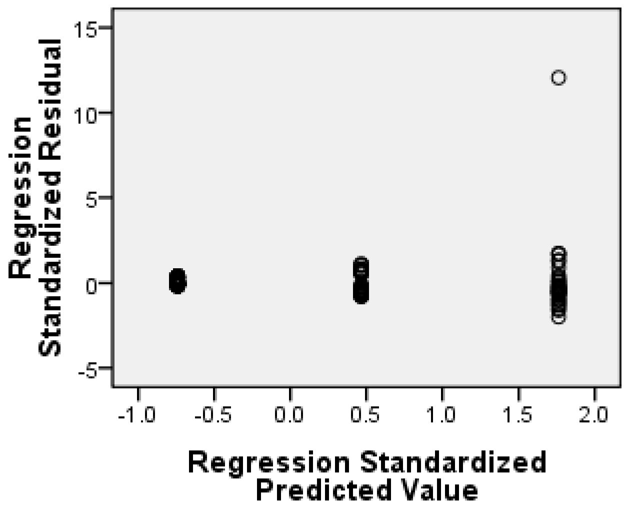


Figure B.6: Residuals against predicted values plot for the processing time model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ◦

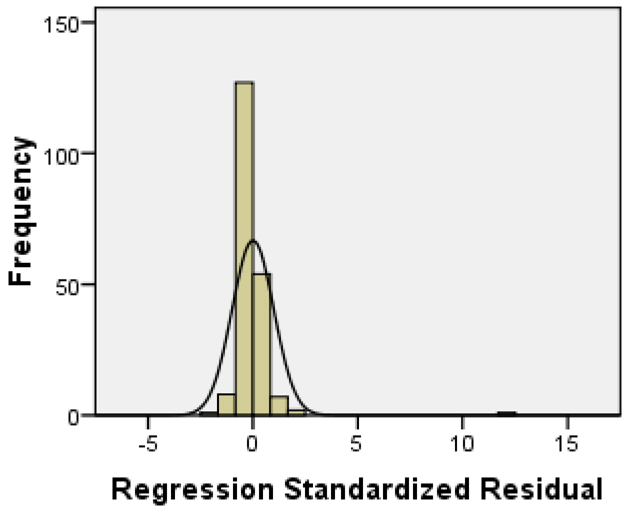


Figure B.7: Histogram of the standardized residuals with normal curve for the processing time model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The curve is not approximated well. ◦

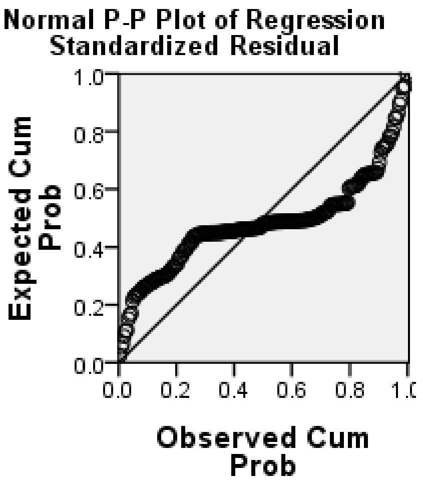


Figure B.8: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The points are clearly off the straight line, indicating deviation from the normal distribution. $\circ$

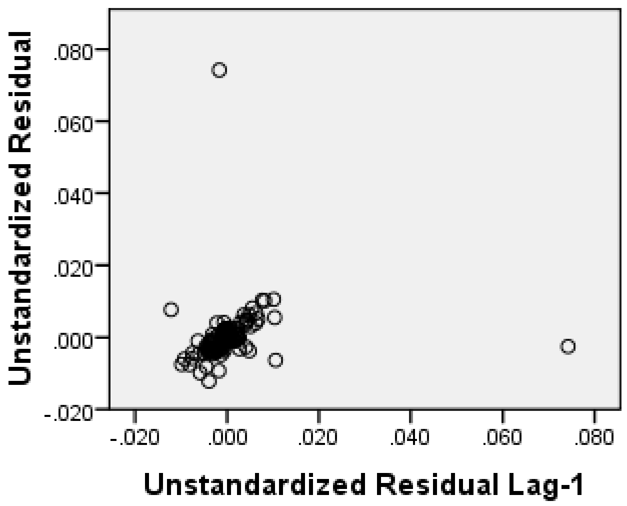


Figure B.9: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The plot indicates little autocorrelation, which is in line with the Durbin-Watson statistic towards 2. $\circ$

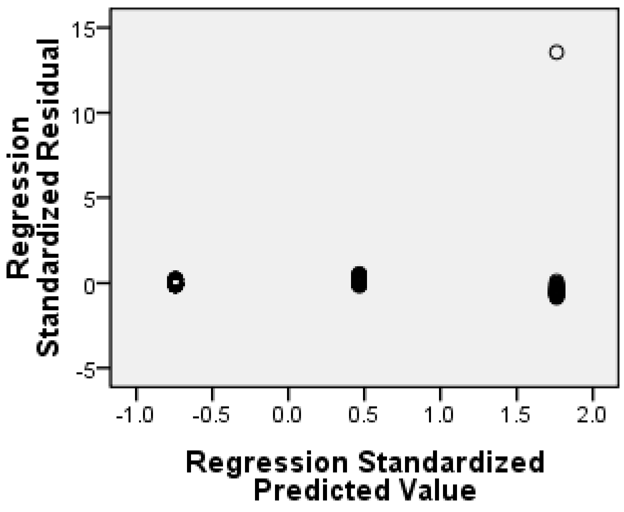


Figure B.10: Residuals against predicted values plot for the processing time model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

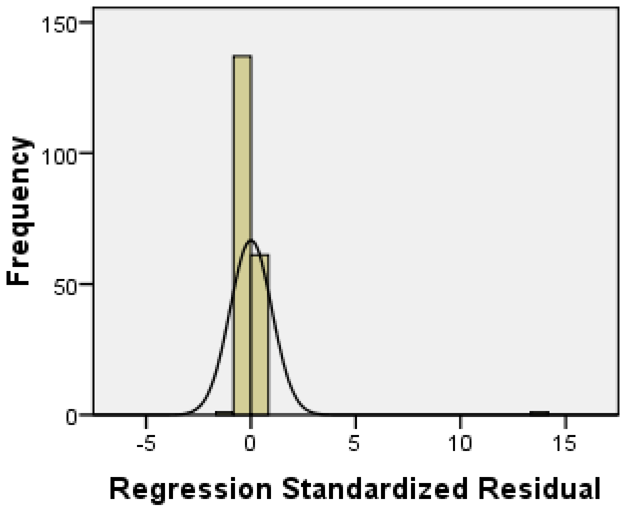


Figure B.11: Histogram of the standardized residuals with normal curve for the processing time model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The curve is not approximated well. ∘

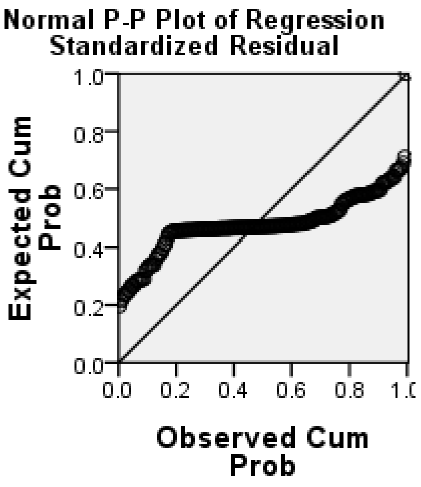


Figure B.12: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The points are clearly off the straight line, indicating deviation from the normal distribution. ◦

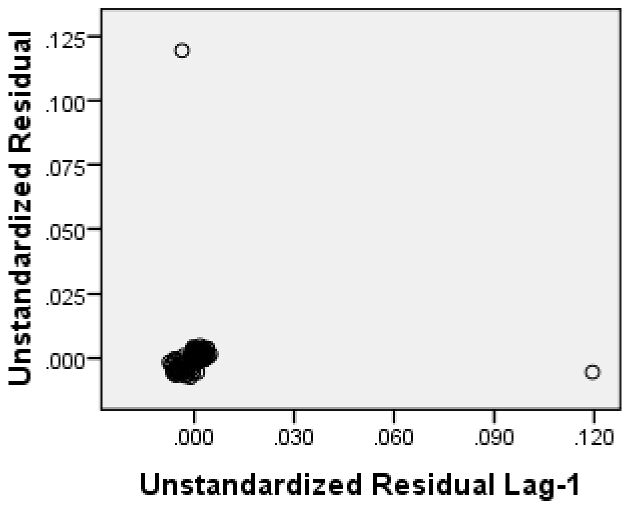


Figure B.13: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The plot indicates little autocorrelation, which is in line with the Durbin-Watson statistic towards 2. ◦

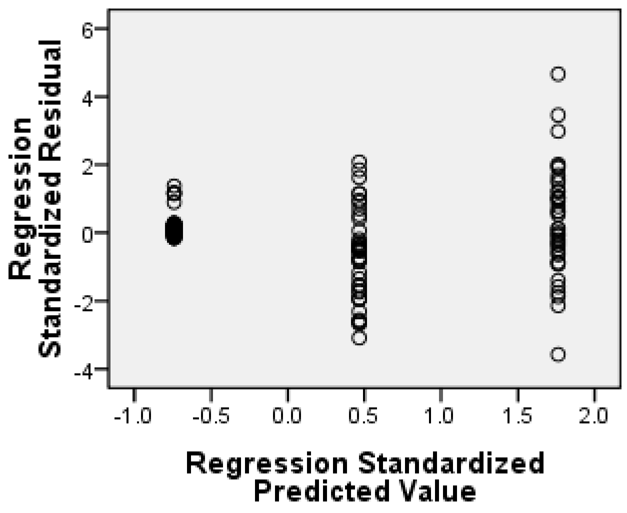


Figure B.14: Residuals against predicted values plot for the processing time model of expiration lists. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ◦

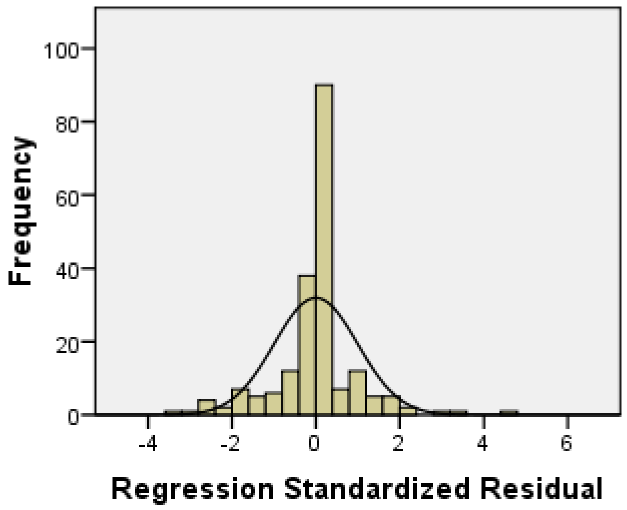


Figure B.15: Histogram of the standardized residuals with normal curve for the processing time model of expiration lists. The curve is not approximated well. ◦

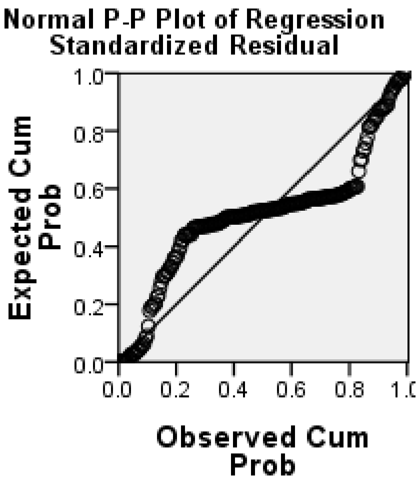


Figure B.16: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of expiration lists. The points are clearly off the straight line, indicating deviation from the normal distribution. ∘

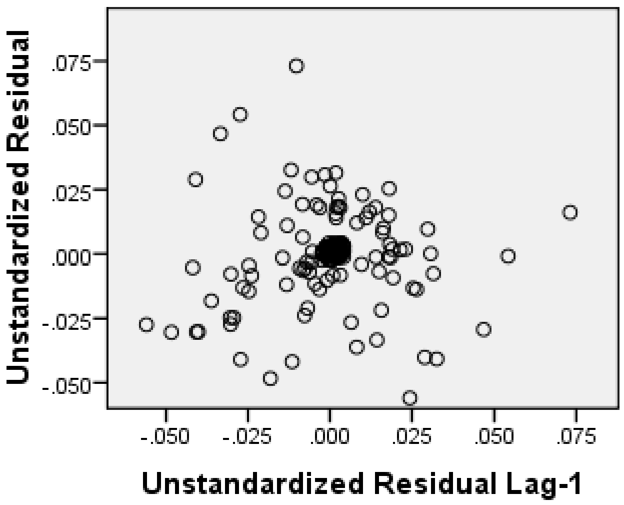


Figure B.17: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of expiration lists. The plot indicates little autocorrelation, which is in line with the Durbin-Watson statistic towards 2. ∘

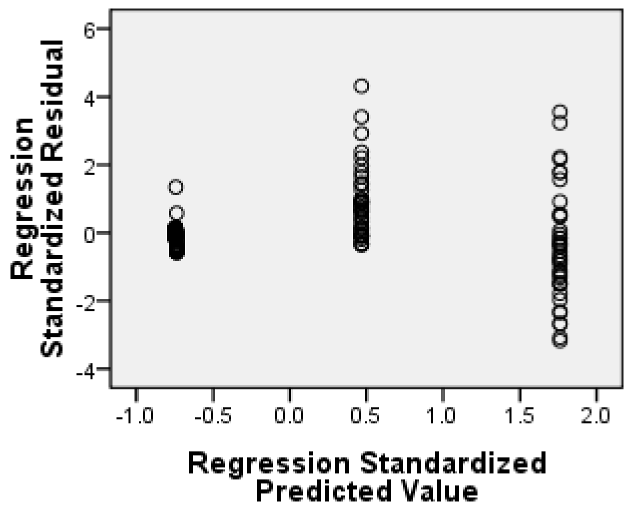


Figure B.18: Residuals against predicted values plot for the processing time model of panes. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

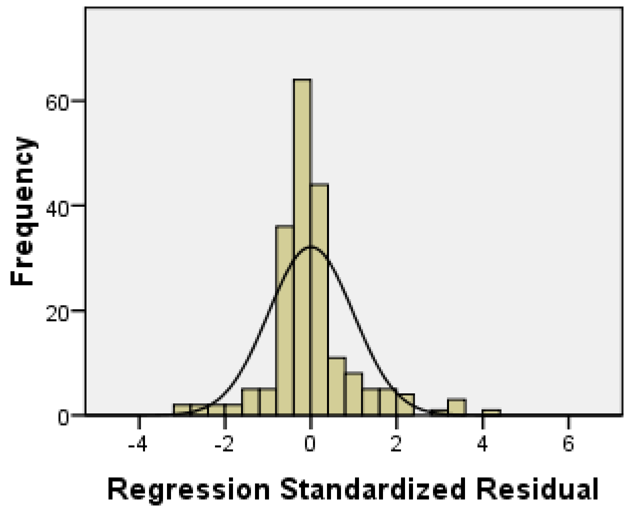


Figure B.19: Histogram of the standardized residuals with normal curve for the processing time model of panes. The curve is not approximated well. ∘

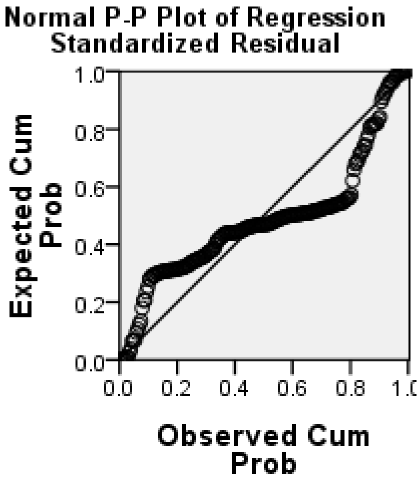


Figure B.20: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of panes. The points are clearly off the straight line, indicating deviation from the normal distribution. ∘

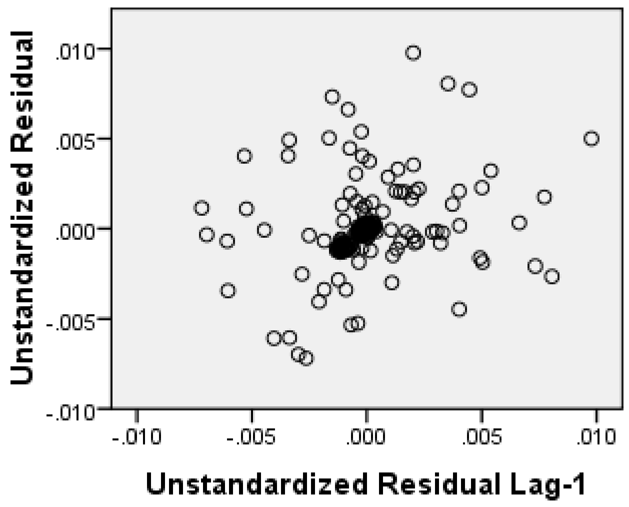


Figure B.21: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of panes. The plot indicates little (positive) autocorrelation, which is in line with the Durbin-Watson statistic towards 2. ∘

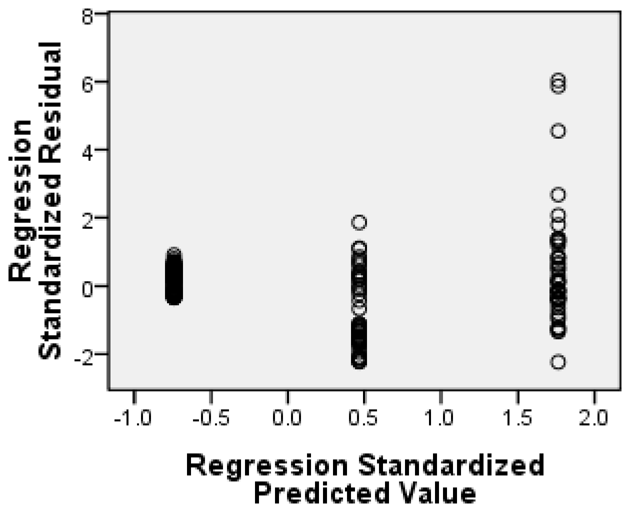


Figure B.22: Residuals against predicted values plot for the processing time model of variance histograms with $\epsilon = 0.05$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

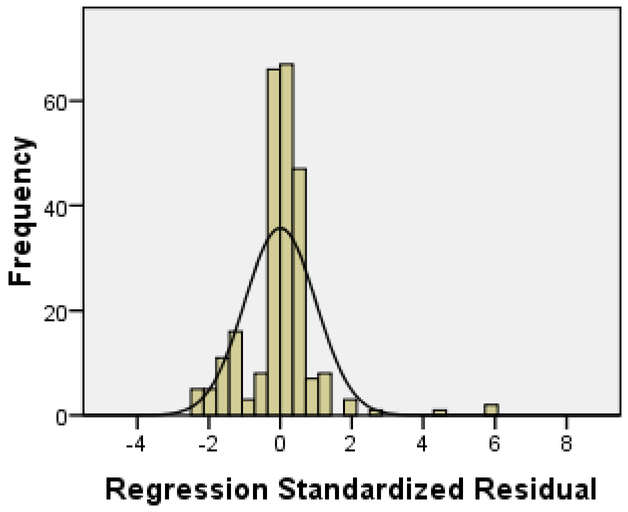


Figure B.23: Histogram of the standardized residuals with normal curve for the processing time model of variance histograms with $\epsilon = 0.05$. The curve is not approximated well. ∘

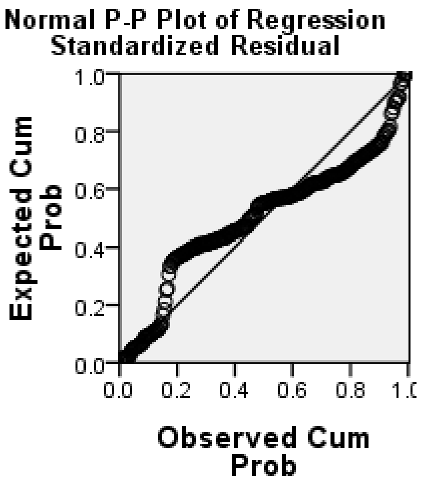


Figure B.24: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of variance histograms with $\epsilon = 0.05$. The points are clearly off the straight line, indicating deviation from the normal distribution. ∘

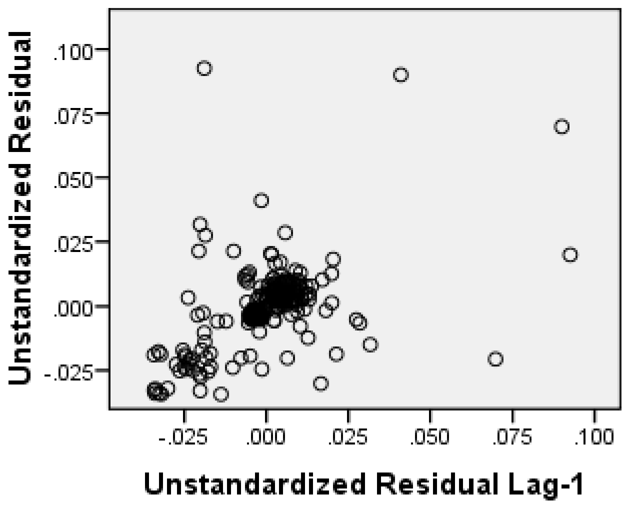


Figure B.25: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of variance histograms with $\epsilon = 0.05$. The plot indicates some positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ∘

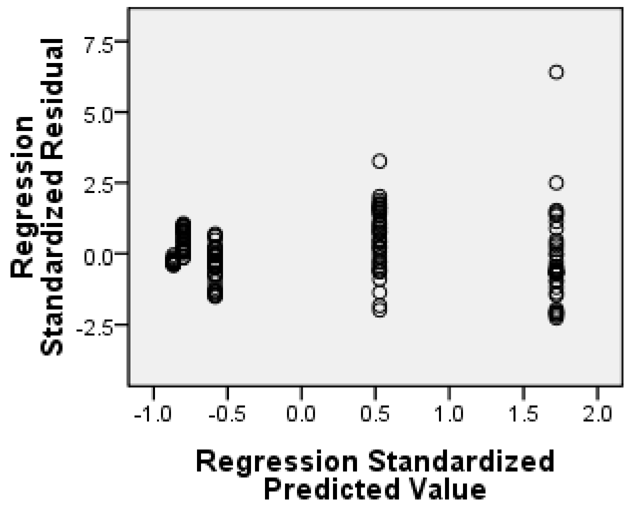


Figure B.26: Residuals against predicted values plot for the processing time model of variance histograms with $\epsilon = 0.01$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. $\circ$

B.2 Memory Space Usage: Variance

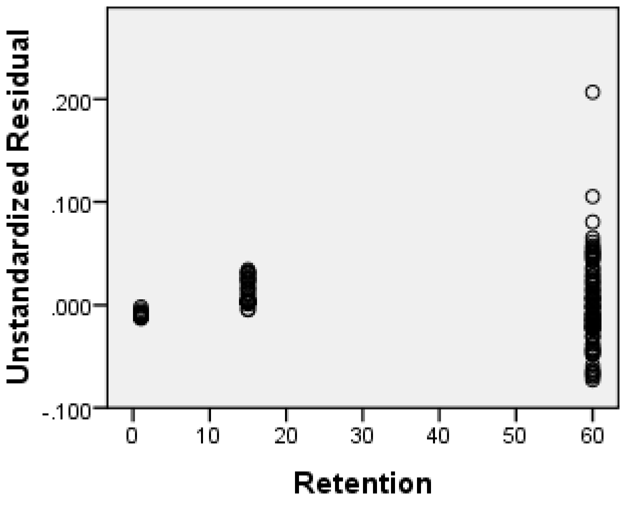


Figure B.27: Residuals against retention plot for the processing time model of variance histograms with $\epsilon = 0.01$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ◦

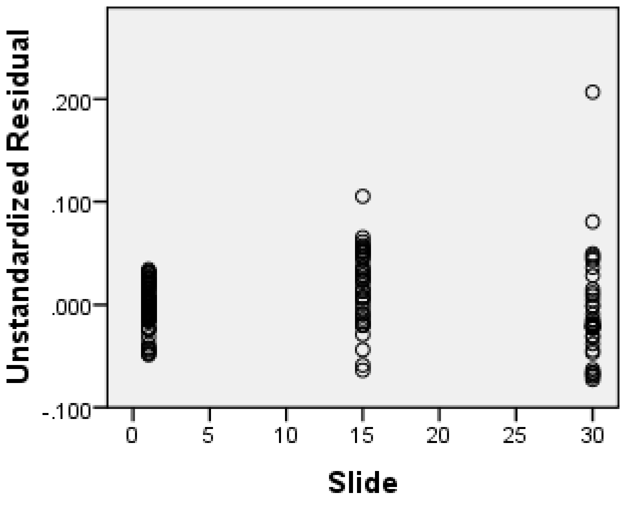


Figure B.28: Residuals against slide plot for the processing time model of variance histograms with $\epsilon = 0.01$. The spread of the residuals shows a slight funnel pattern, indicating heteroscedasticity. ◦

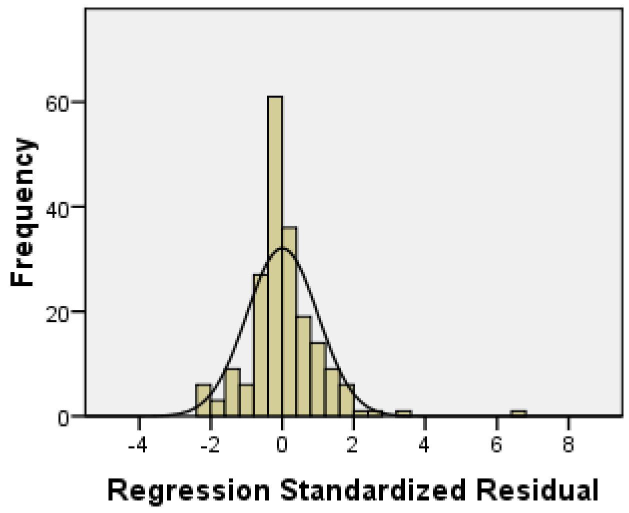


Figure B.29: Histogram of the standardized residuals with normal curve for the processing time model of variance histograms with $\epsilon = 0.01$. The curve is approximated reasonably. ∘

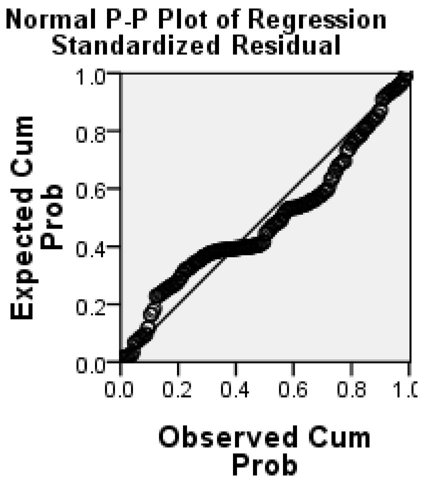


Figure B.30: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of variance histograms with $\epsilon = 0.01$. The points are slightly off the straight line, indicating deviation from the normal distribution. ∘

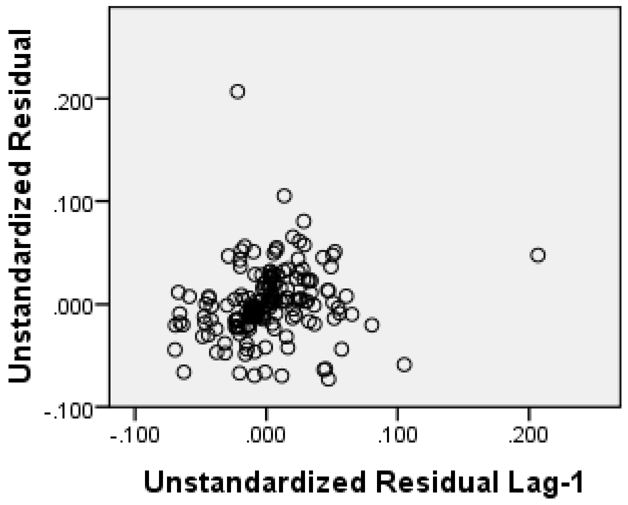


Figure B.31: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of variance histograms with $\epsilon = 0.01$. The plot indicates hardly any (positive) autocorrelation, which is in line with the Durbin-Watson statistic towards 2. $\circ$

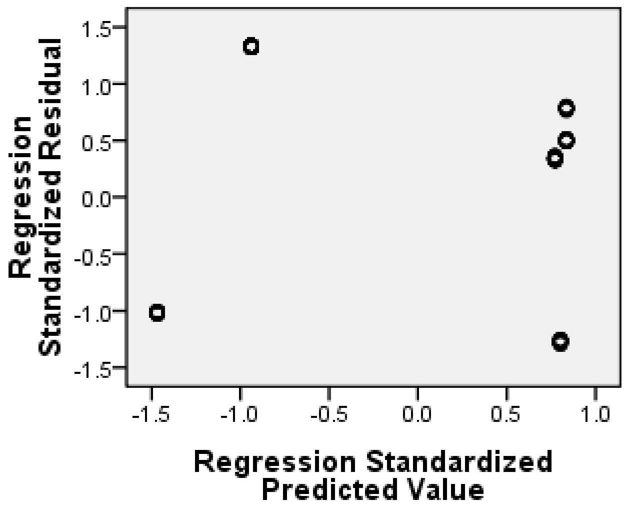


Figure B.32: Residuals-predicted plot for the memory space usage model of the instantaneous approach. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. $\circ$

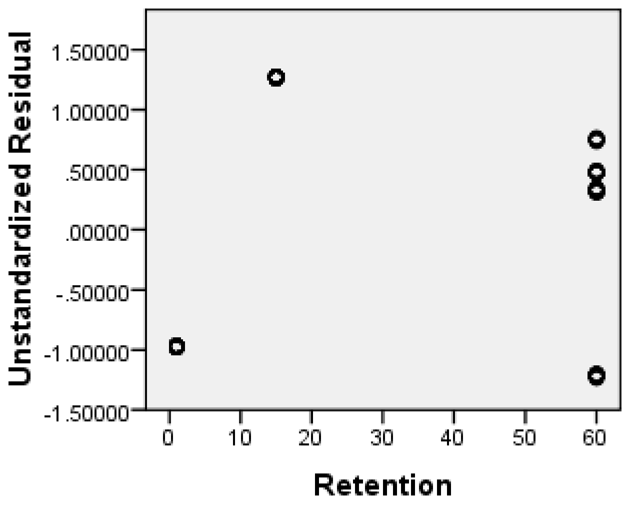


Figure B.33: Residuals-retention plot for the memory space usage model of the instantaneous approach. The spread of the residuals depends on the values for retention, indicating heteroscedasticity. ◦

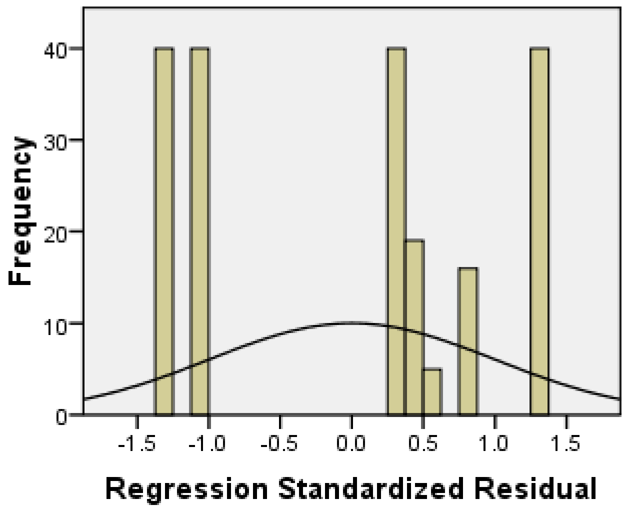


Figure B.34: Histogram of the standardized residuals with normal curve for the memory space usage model of the instantaneous approach. The curve is not approximated well. ◦

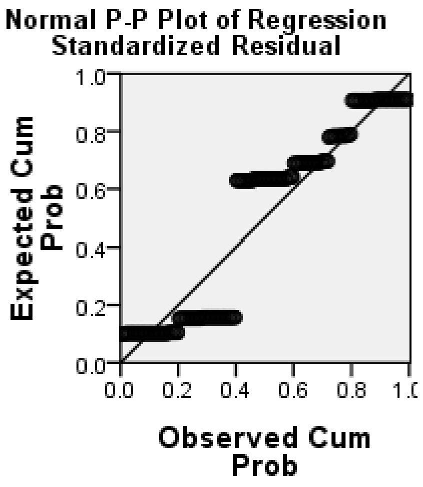


Figure B.35: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the instantaneous approach. The points are clearly off the straight line, indicating deviation from the normal distribution. ○

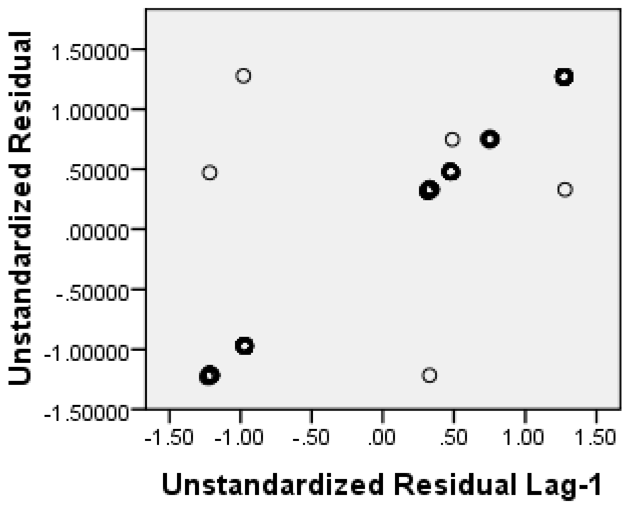


Figure B.36: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the instantaneous approach. The linear relationship indicates very strong positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ○

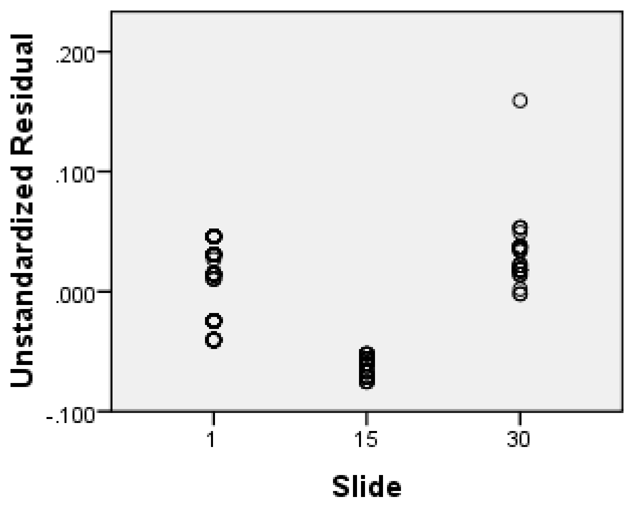


Figure B.37: Residuals-slide plot for the memory space usage model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The spread of the residuals depends on the value of slide, indicating heteroscedasticity. ∘

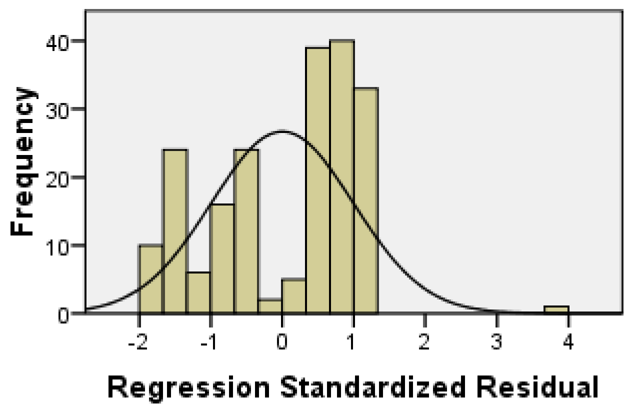


Figure B.38: Histogram of the standardized residuals with normal curve for the memory space usage model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The curve is not approximated well. ∘

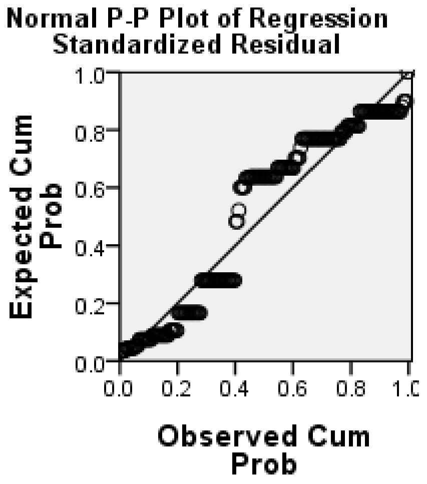


Figure B.39: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The points are clearly off the straight line, indicating deviation from the normal distribution. ○

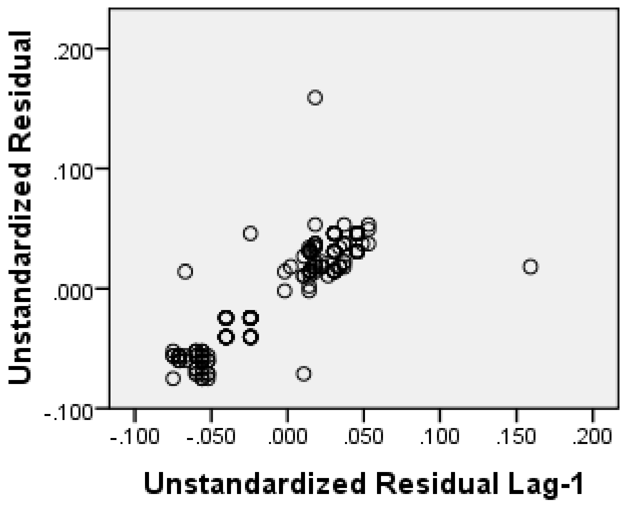


Figure B.40: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The linear relationship indicates positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ○

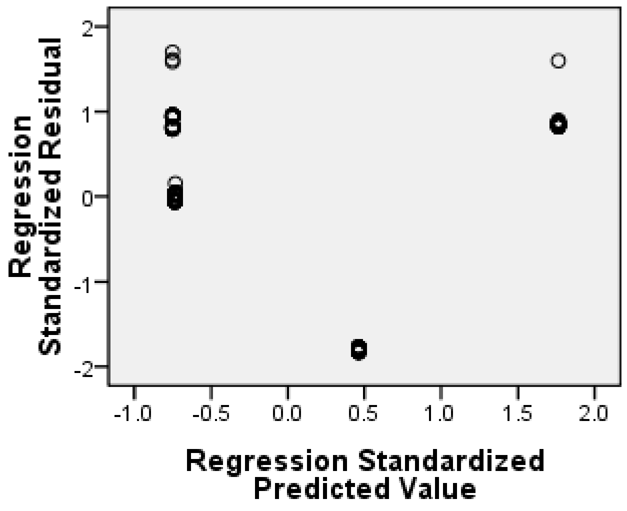


Figure B.41: Residuals-predicted plot for the memory space usage model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ◦

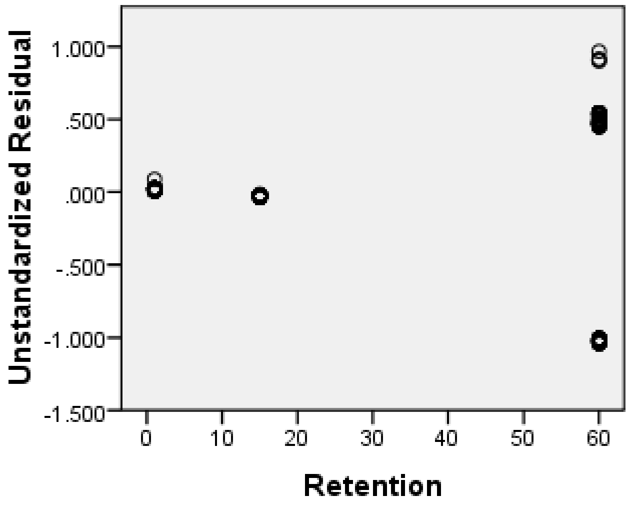


Figure B.42: Residuals-retention plot for the memory space usage model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals depends on the values for retention, indicating heteroscedasticity. ◦

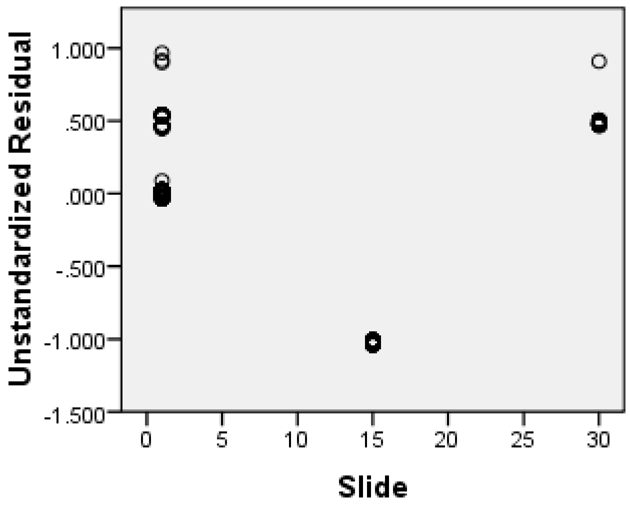


Figure B.43: Residuals-slide plot for the memory space usage model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals depends on the values for slide, indicating heteroscedasticity. ∘

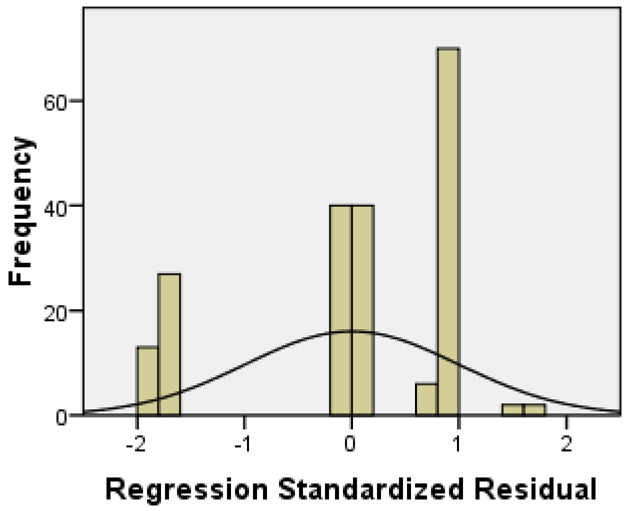


Figure B.44: Histogram of the standardized residuals with normal curve for the memory space usage model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The curve is not approximated well. ∘

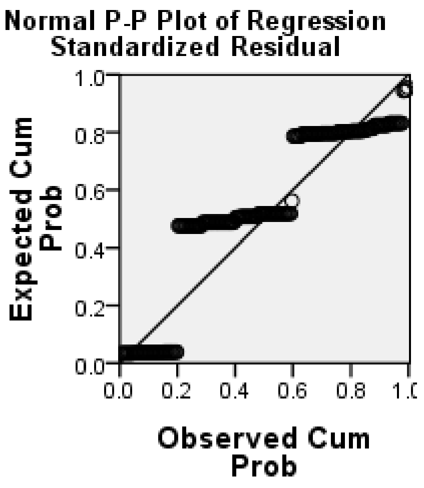


Figure B.45: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The points are clearly off the straight line, indicating deviation from the normal distribution. ◦

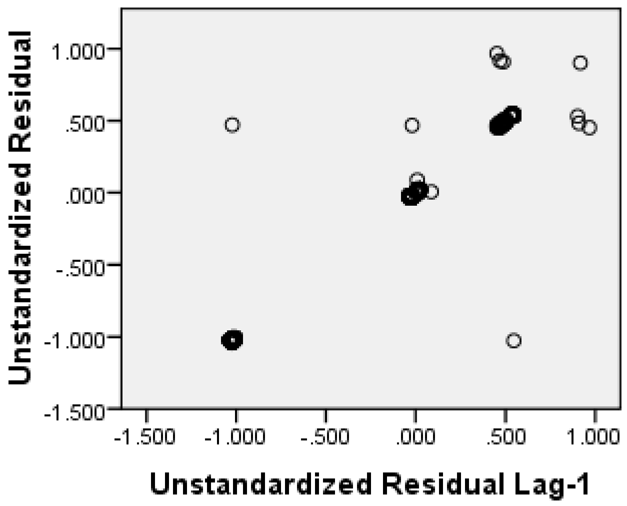


Figure B.46: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The linear relationship indicates positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ◦

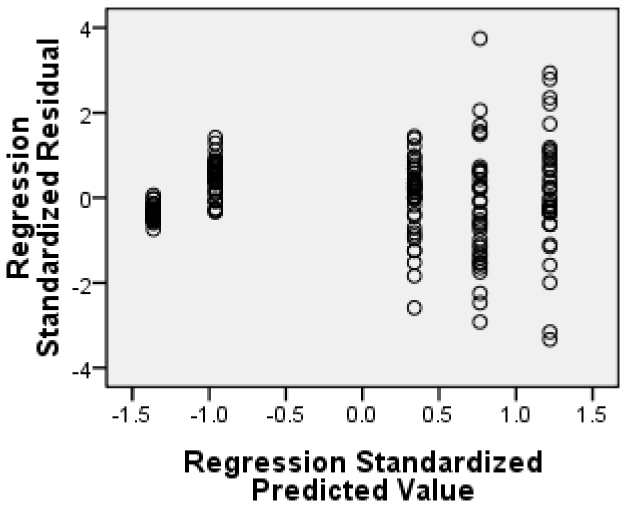


Figure B.47: Residuals-predicted plot for the memory space usage model of expiration lists. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ◦

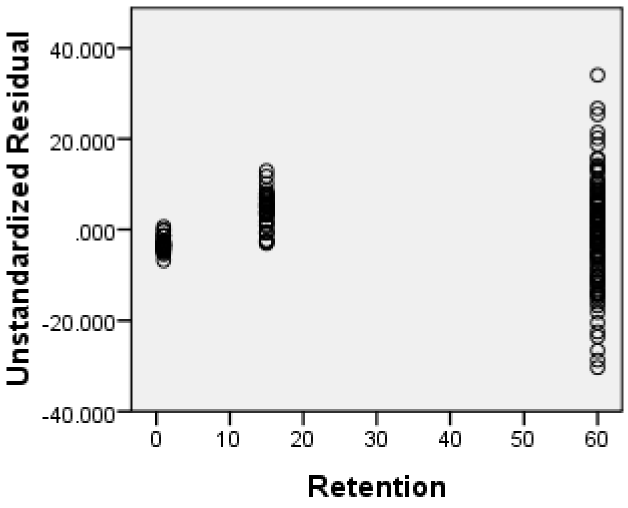


Figure B.48: Residuals-retention plot for the memory space usage model of expiration lists. The spread of the residuals depends on the values for retention, indicating heteroscedasticity. ◦

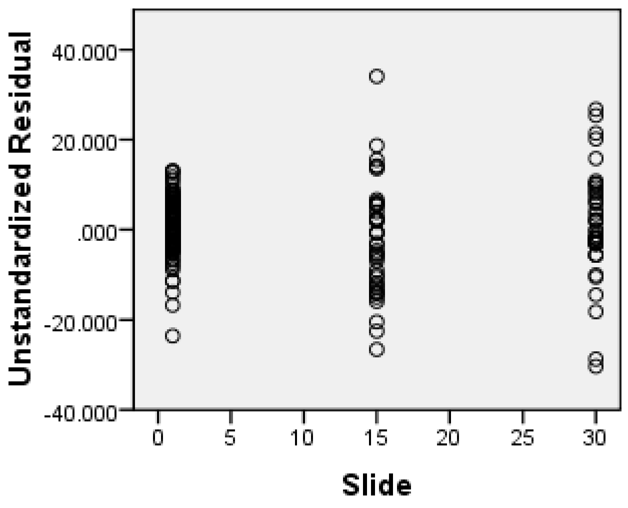


Figure B.49: Residuals-slide plot for the memory space usage model of expiration lists. The spread of the residuals depends on the values for slide, indicating heteroscedasticity. ◦

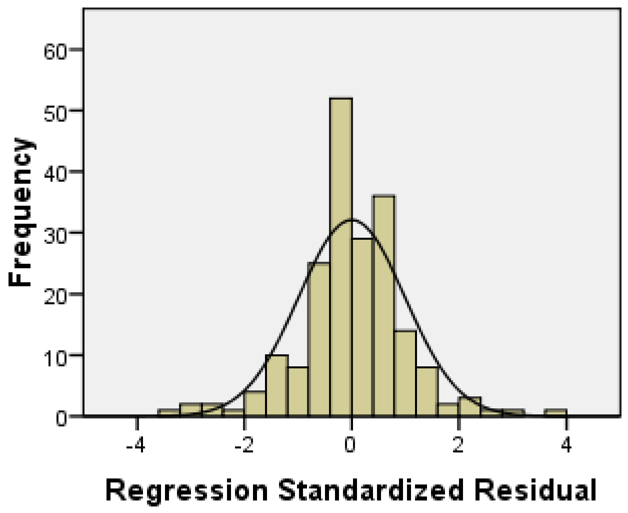


Figure B.50: Histogram of the standardized residuals with normal curve for the memory space usage model of expiration lists. The curve is approximated well. ◦

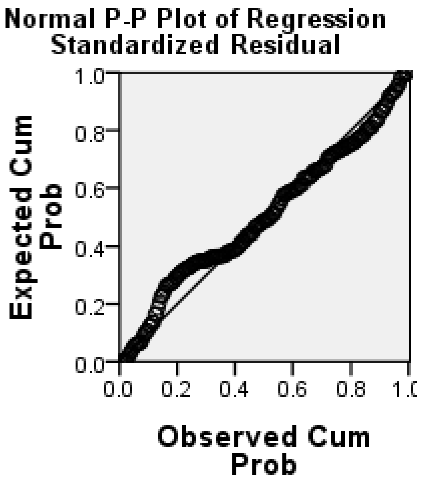


Figure B.51: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of expiration lists. The points approximate the straight line, indicating an approximate normal distribution. ◦

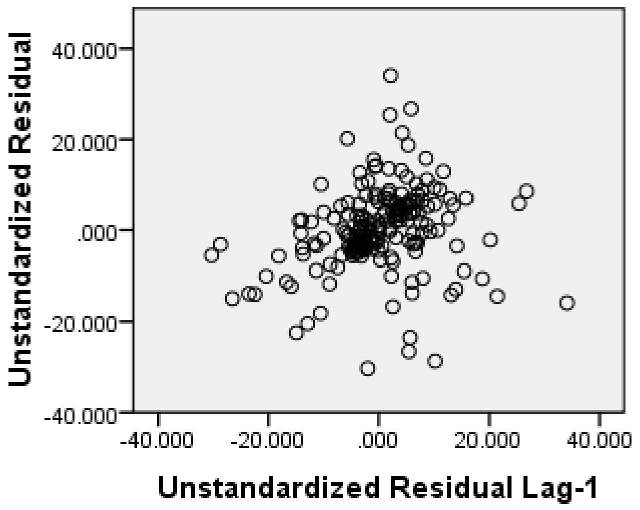


Figure B.52: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of expiration lists. The linear relationship indicates slight positive autocorrelation, which is in line with the Durbin-Watson statistic towards 2. ◦

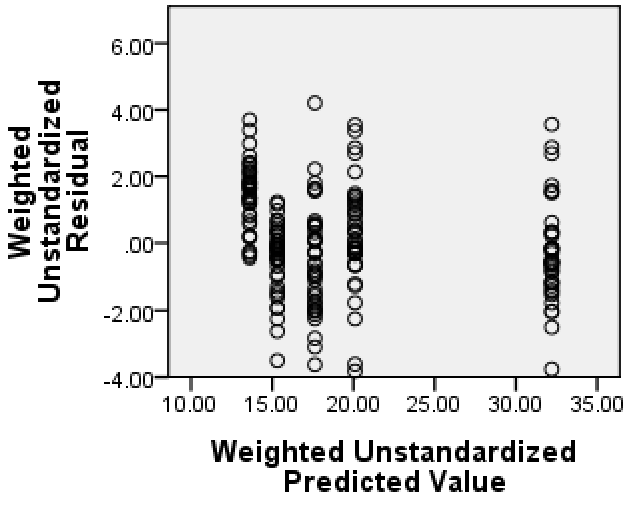


Figure B.53: Weighted residuals - weighted predicted plot for the weighted memory space usage model of expiration lists. The spread of the residuals varies slightly with the predicted values, indicating mild heteroscedasticity. ◦

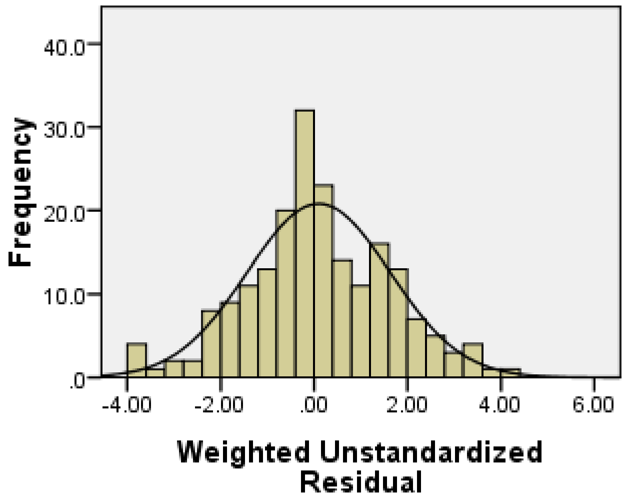


Figure B.54: Histogram of the weighted residuals with normal curve for the weighted memory space usage model of expiration lists. The curve is approximated well. ◦

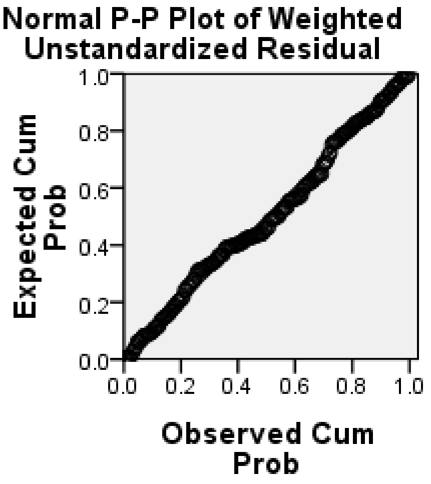


Figure B.55: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) weighted residuals distribution for the memory space usage model of expiration lists. The points approximate the straight line, indicating an approximate normal distribution. ∘

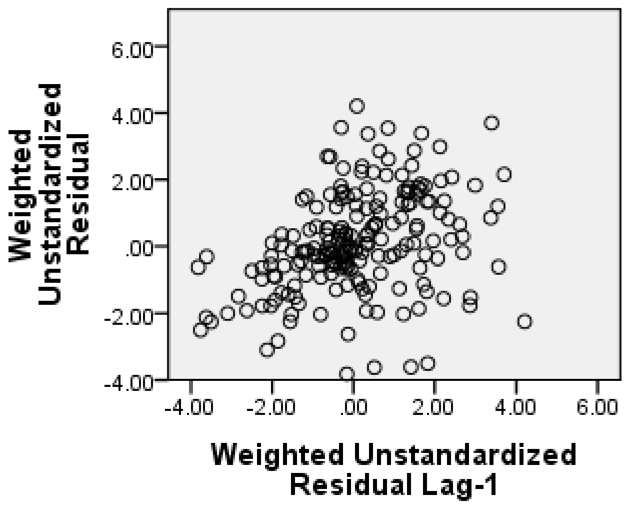


Figure B.56: Lag plot of weighted unstandardized residuals e_i against lag 1 weighted unstandardized residuals e_{i-1} for the memory space usage model of expiration lists. The linear relationship indicates slight positive autocorrelation. ∘

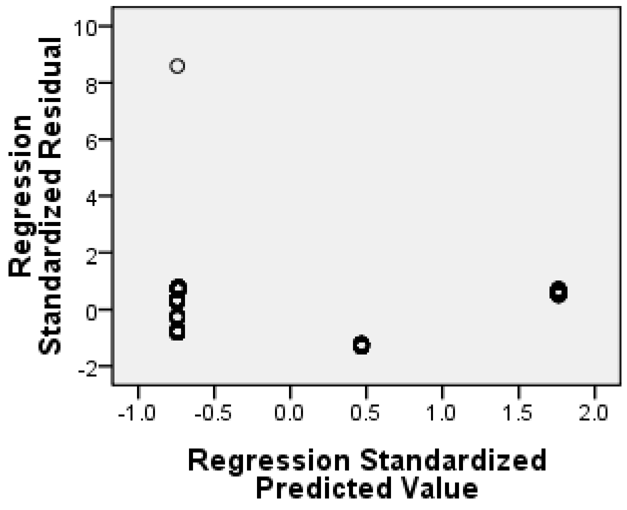


Figure B.57: Residuals-predicted plot for the memory space usage model of the incremental approach using panes. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ○

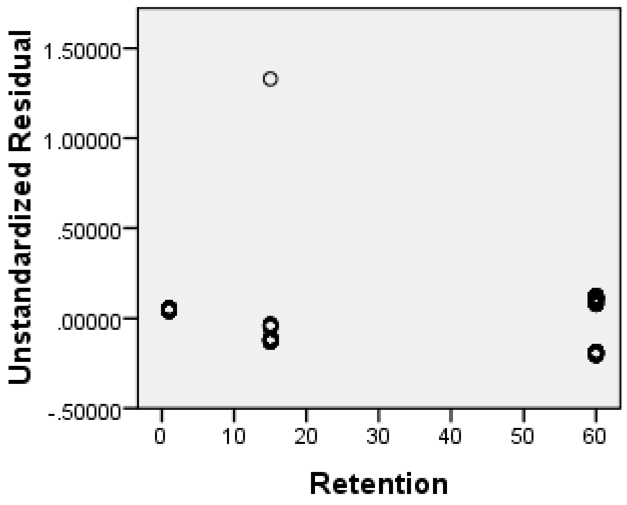


Figure B.58: Residuals-retention plot for the memory space usage model of the incremental approach using panes. The spread of the residuals varies with the values for retention, indicating heteroscedasticity. ○

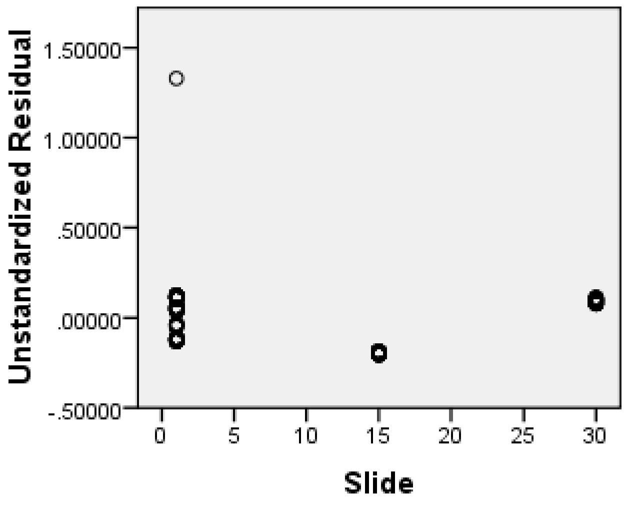


Figure B.59: Residuals-slide plot for the memory space usage model of the incremental approach using panes. The spread of the residuals varies with the values for slide, indicating heteroscedasticity. ○

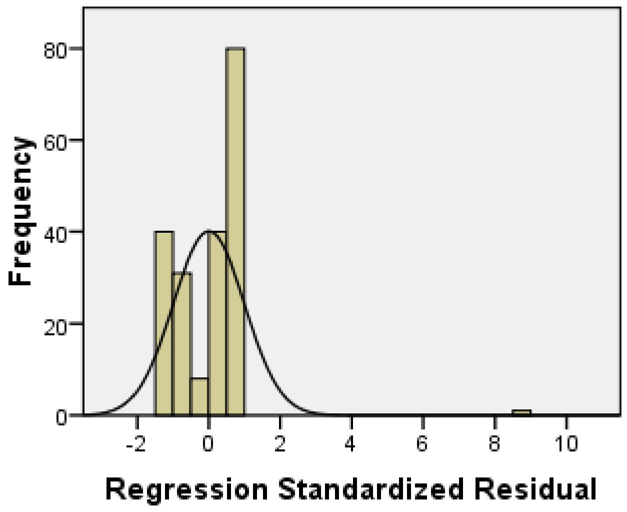


Figure B.60: Histogram of the standardized residuals with normal curve for the memory space usage model of the incremental approach with panes. The curve is not approximated well. ○

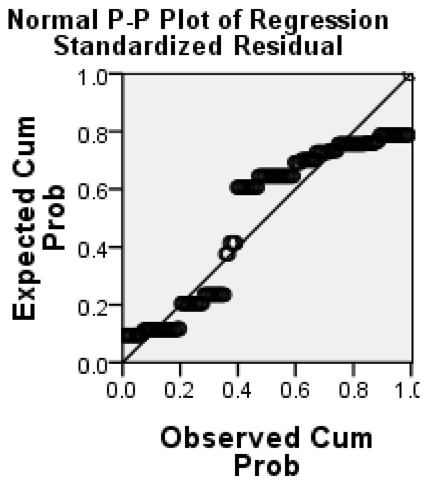


Figure B.61: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the incremental approach using panes. The points are clearly off the straight line, indicating deviation from the normal distribution. ○

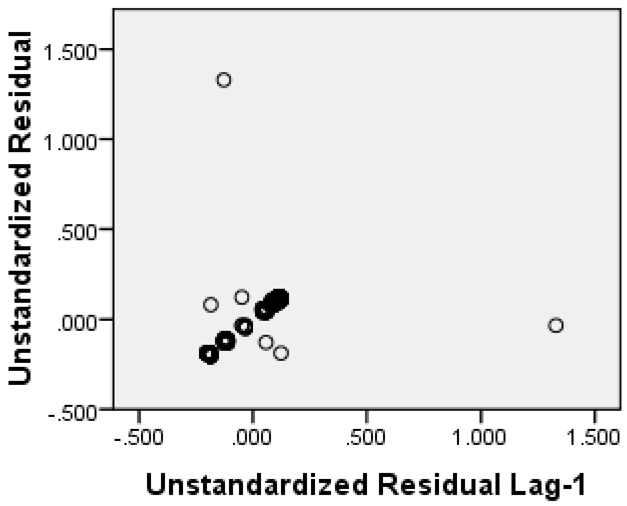


Figure B.62: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the incremental approach using panes. The linear relationship indicates clear positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ○

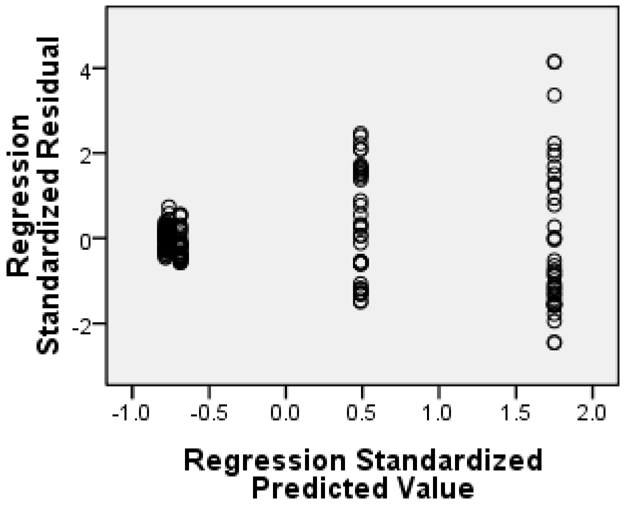


Figure B.63: Residuals-predicted plot for the memory space usage model of variance histograms with $\epsilon = 0.05$. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ○

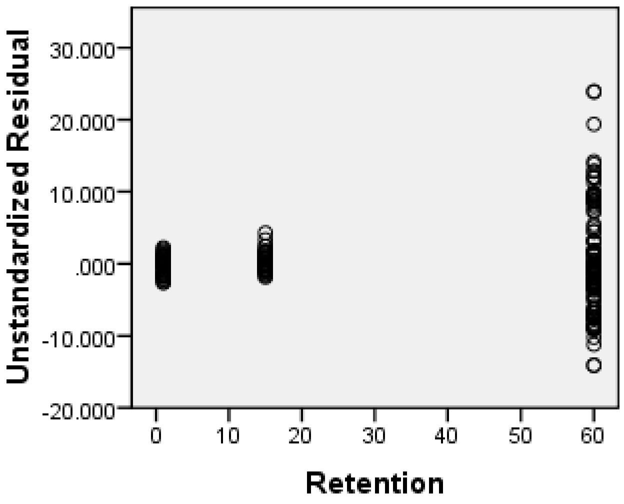


Figure B.64: Residuals-retention plot for the memory space usage model of variance histograms with $\epsilon = 0.05$. The spread of the residuals depends on the values for retention, indicating heteroscedasticity. ○

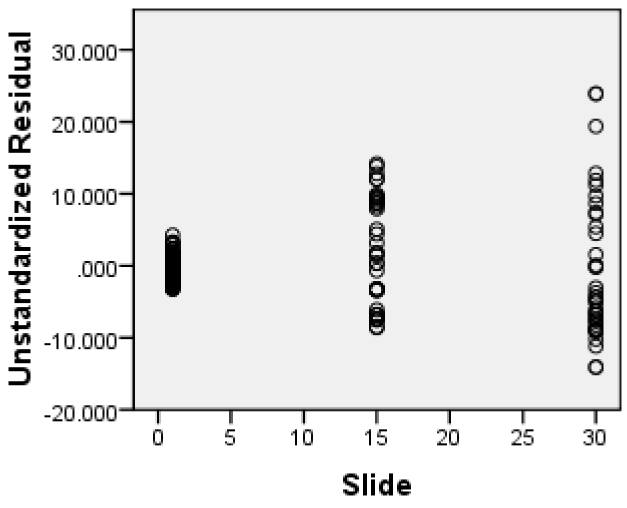


Figure B.65: Residuals-slide plot for the memory space usage model of variance histograms with $\epsilon = 0.05$. The spread shows a funnel pattern, indicating heteroscedasticity. ○

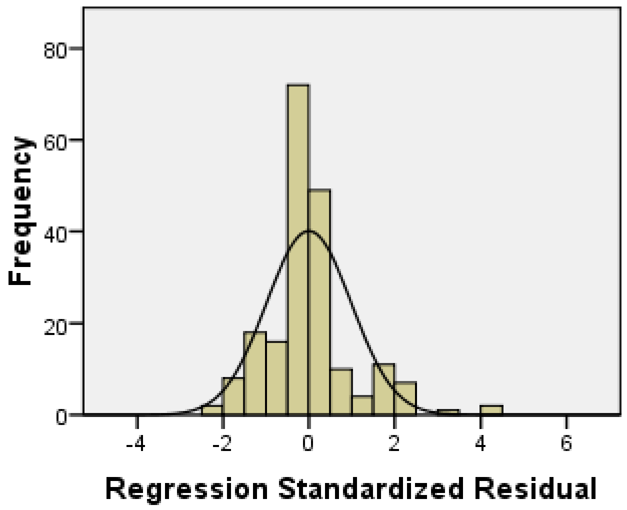


Figure B.66: Histogram of the standardized residuals with normal curve for the memory space usage model of variance histograms with $\epsilon = 0.05$. The curve is approximated well. ○

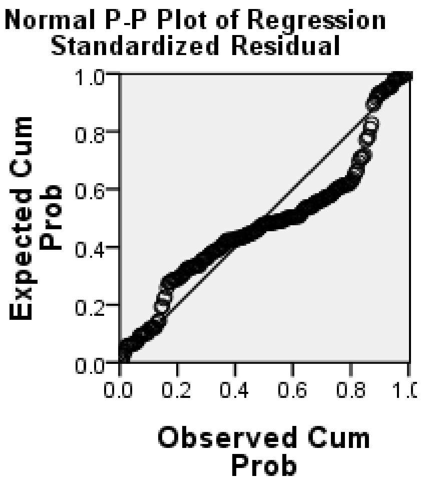


Figure B.67: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of variance histograms with $\epsilon = 0.05$. The points approximate the straight line, indicating an approximate normal distribution. ∘

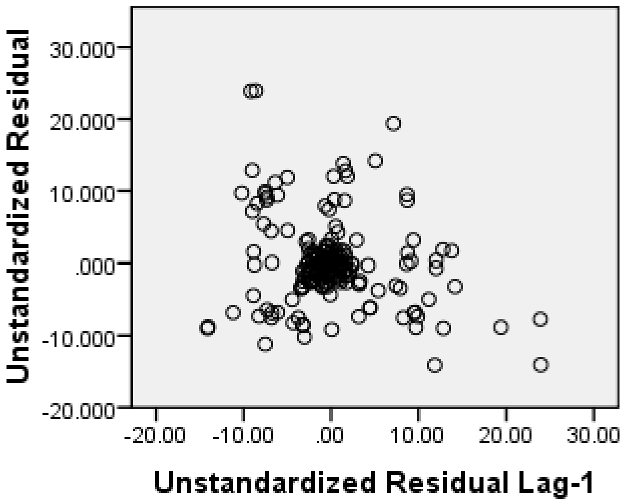


Figure B.68: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of variance histograms with $\epsilon = 0.05$. The linear relationship indicates slight negative autocorrelation, which is in line with the Durbin-Watson statistic towards 4. ∘

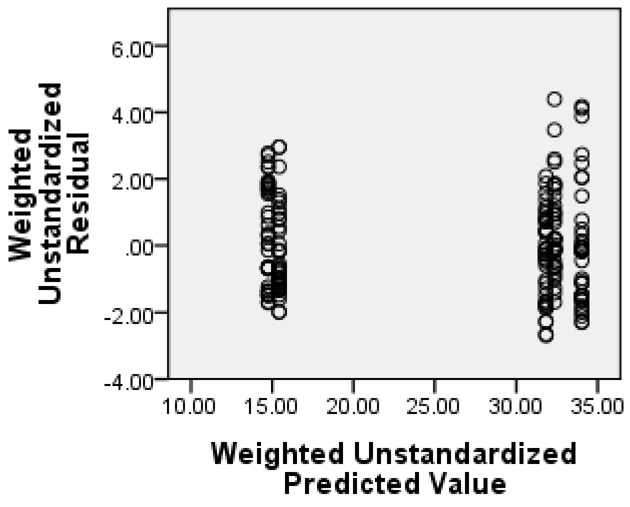


Figure B.69: Weighted residuals - weighted predicted plot for the weighted memory space usage model of variance histograms with $\epsilon = 0.05$. The spread of the residuals varies slightly with the predicted values, indicating mild heteroscedasticity. ∘

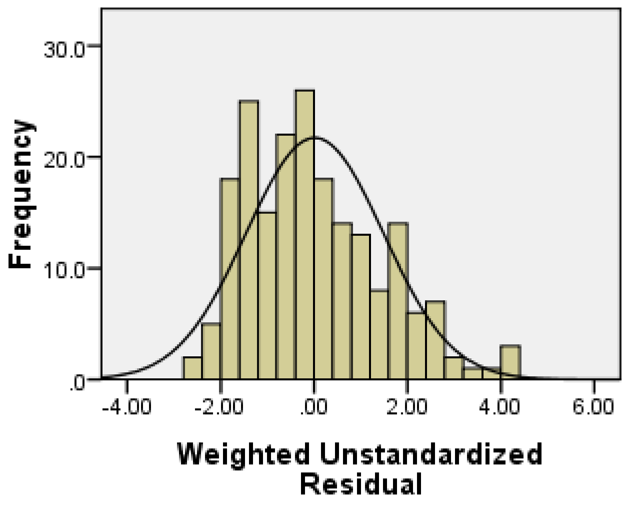


Figure B.70: Histogram of the weighted residuals with normal curve for the weighted memory space usage model of variance histograms with $\epsilon = 0.05$. The curve is approximated well. ∘

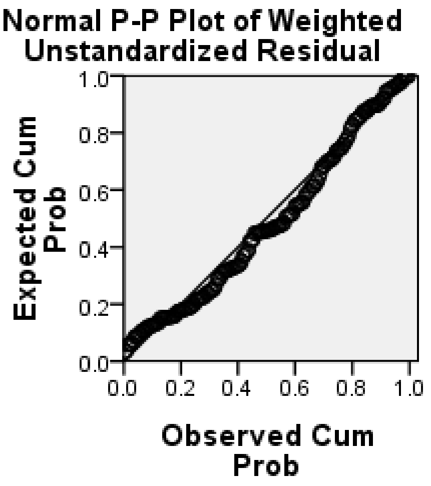


Figure B.71: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) weighted residuals distribution for the memory space usage model of variance histograms with $\epsilon = 0.05$. The points approximate the straight line, indicating an approximate normal distribution. ∘

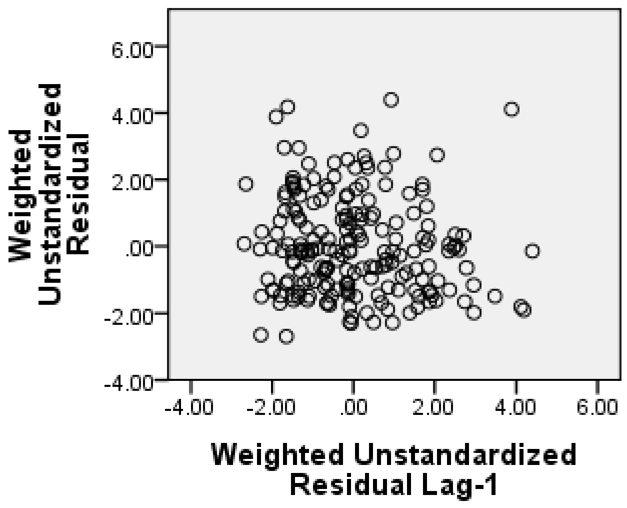


Figure B.72: Lag plot of weighted unstandardized residuals e_i against lag 1 weighted unstandardized residuals e_{i-1} for the memory space usage model of variance histograms with $\epsilon = 0.05$. The plot indicates hardly any autocorrelation. ∘

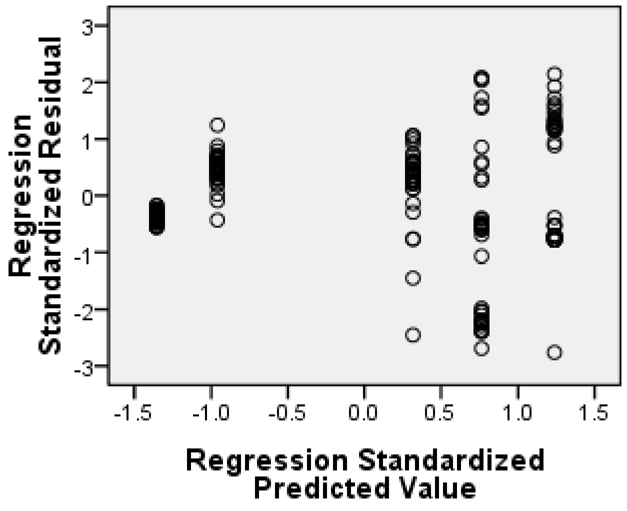


Figure B.73: Residuals-predicted plot for the memory space usage model of variance histograms with $\epsilon = 0.01$. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. $\circ$

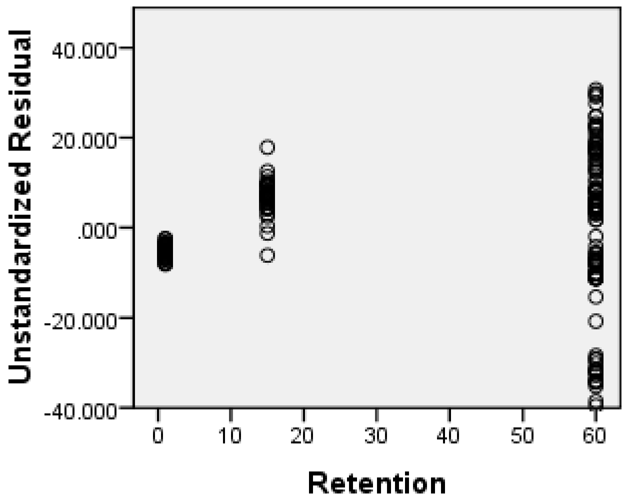


Figure B.74: Residuals-retention plot for the memory space usage model of variance histograms with $\epsilon = 0.01$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. $\circ$

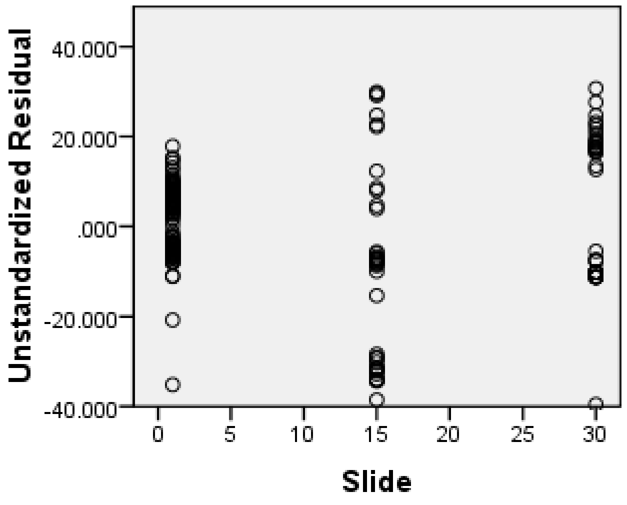


Figure B.75: Residuals-slide plot for the memory space usage model of variance histograms with $\epsilon = 0.01$. The spread shows small variance at low and high values for slide and larger variance at middle values, indicating some heteroscedasticity. ◦

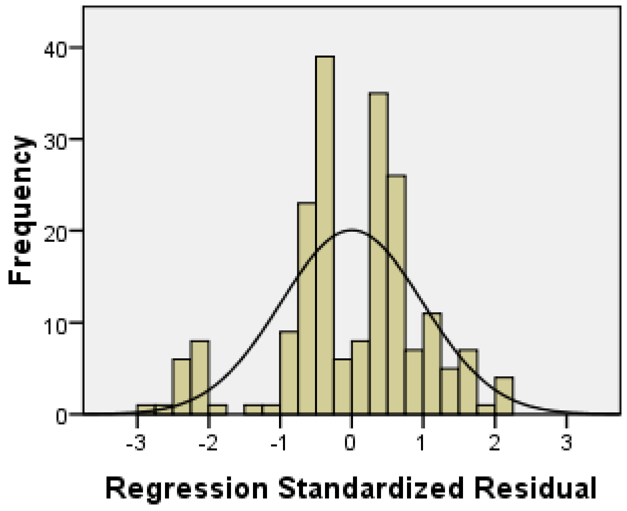


Figure B.76: Histogram of the standardized residuals with normal curve for the memory space usage model of variance histograms with $\epsilon = 0.01$. The curve is approximated well. ◦

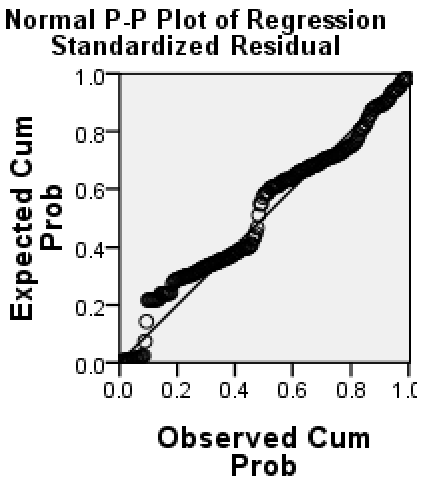


Figure B.77: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of variance histograms with $\epsilon = 0.01$. The points approximate the straight line, indicating an approximate normal distribution. ◦

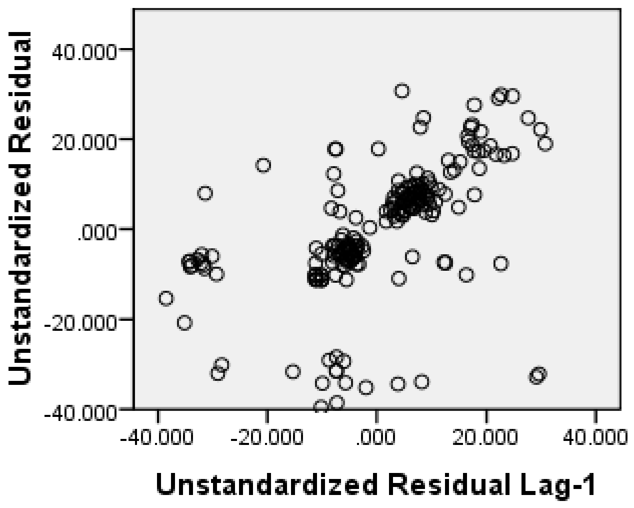


Figure B.78: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of variance histograms with $\epsilon = 0.01$. The linear relationship indicates positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ◦

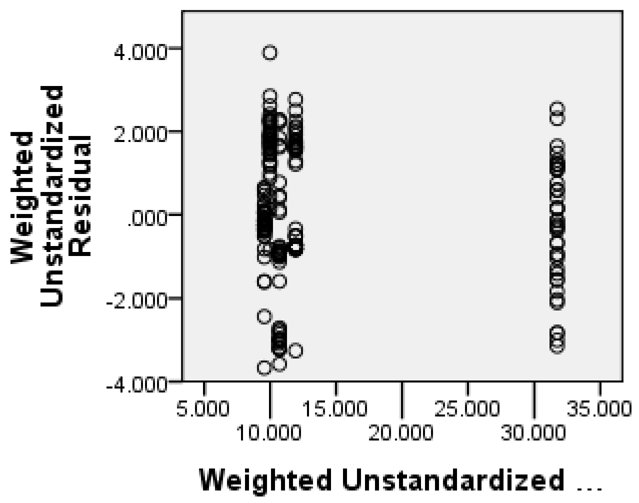


Figure B.79: Weighted residuals - weighted predicted plot for the weighted memory space usage model of variance histograms with $\epsilon = 0.01$. The spread of the residuals varies slightly with the predicted values, indicating mild heteroscedasticity. ∘

B.3 Processing Time: Quantiles

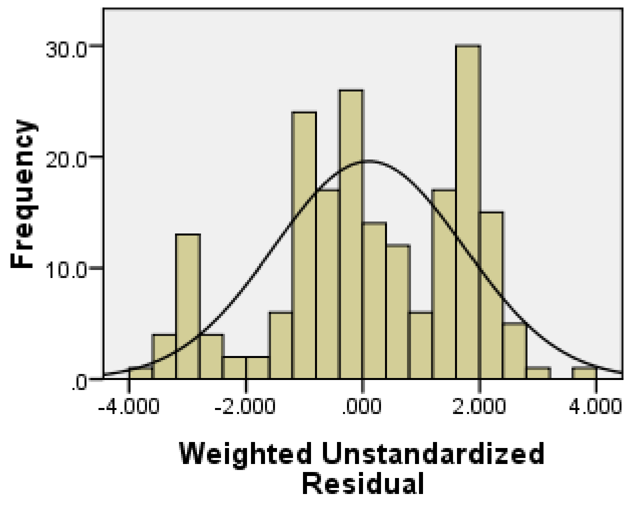


Figure B.80: Histogram of the weighted residuals with normal curve for the weighted memory space usage model of variance histograms with $\epsilon = 0.01$. The curve is approximated not too well. ◦

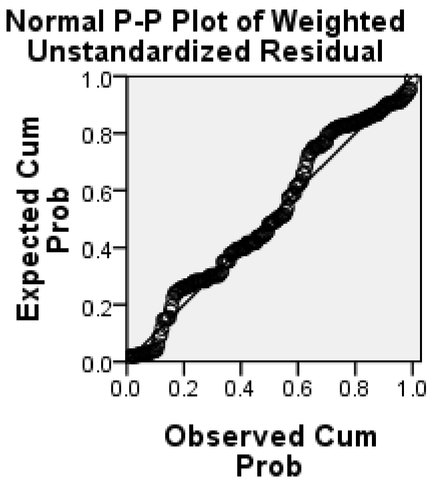


Figure B.81: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) weighted residuals distribution for the memory space usage model of variance histograms with $\epsilon = 0.01$. The points approximate the straight line, indicating an approximate normal distribution. ◦

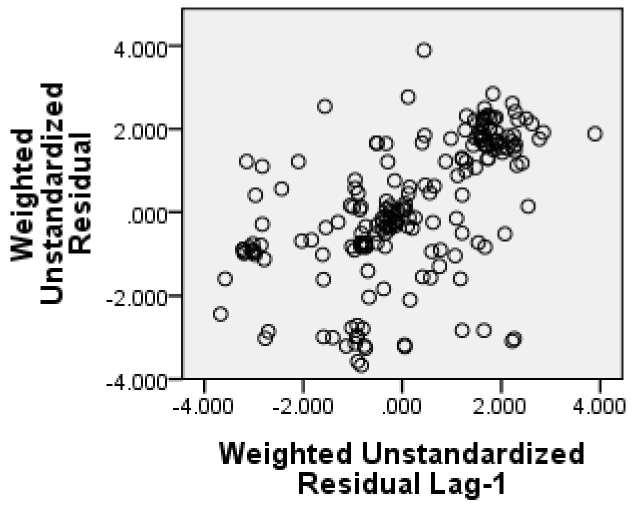


Figure B.82: Lag plot of weighted unstandardized residuals e_i against lag 1 weighted unstandardized residuals e_{i-1} for the memory space usage model of variance histograms with $\epsilon = 0.01$. The linear relationship indicates positive autocorrelation. ∘

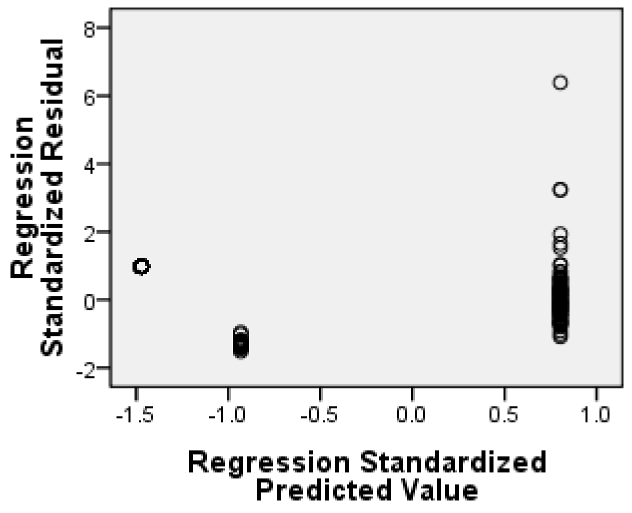


Figure B.83: Residuals against predicted values plot for the processing time model of the instantaneous approach. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

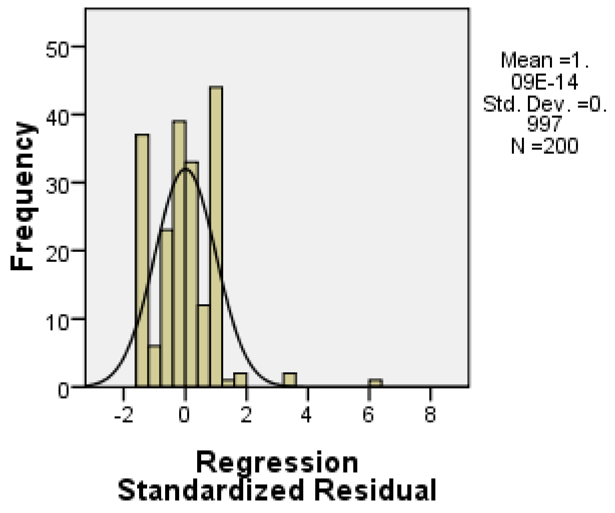


Figure B.84: Histogram of the standardized residuals with normal curve for the processing time model of the instantaneous approach. The curve is not approximated well. ◦

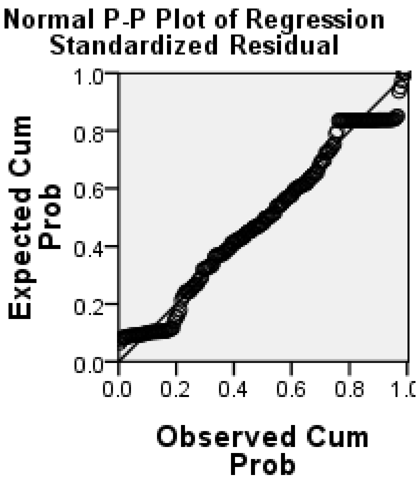


Figure B.85: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the instantaneous approach. The points approximate a straight line, indicating some deviation from the normal distribution. ◦

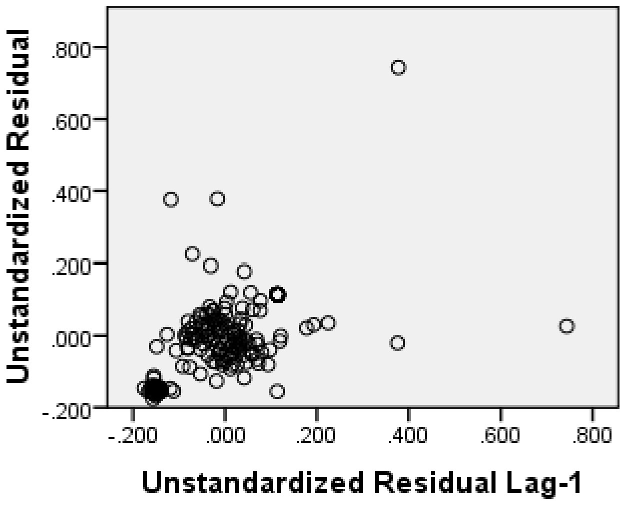


Figure B.86: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the instantaneous approach. The plot indicates positive autocorrelation, which is in line with the Durbin-Watson statistic towards 1. $\circ$

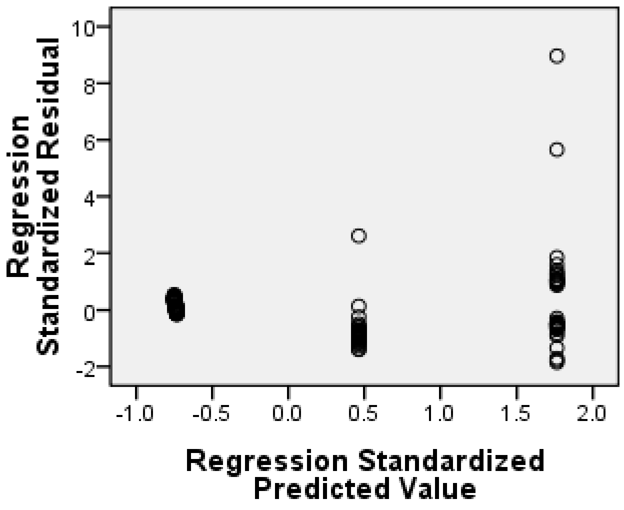


Figure B.87: Residuals against predicted values plot for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ∘

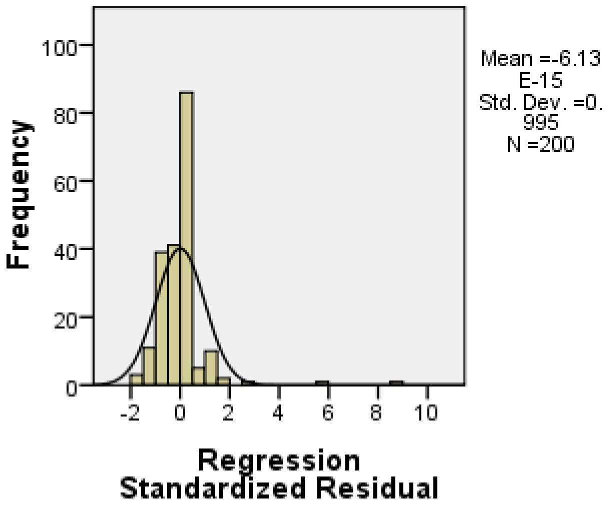


Figure B.88: Histogram of the standardized residuals with normal curve for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The curve is approximated. ∘

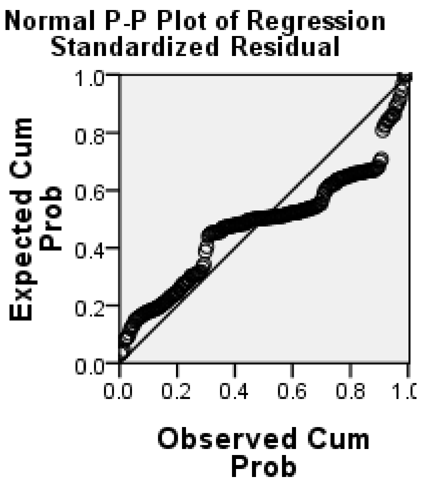


Figure B.89: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The points approximate the straight line, indicating an approximate normal distribution. ◦

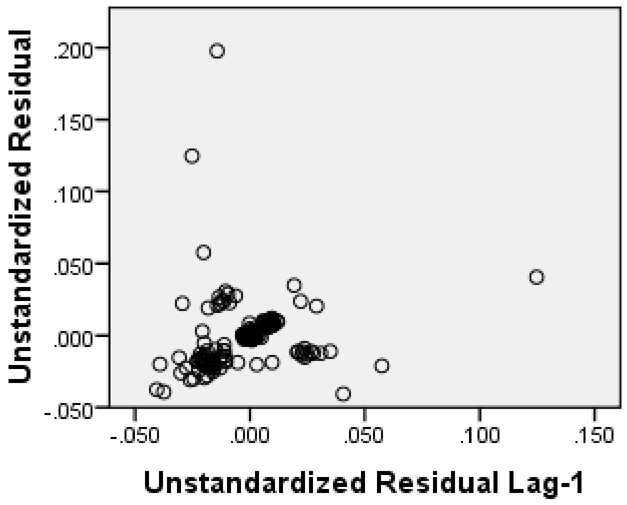


Figure B.90: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The plot indicates positive autocorrelation, which is in line with the Durbin-Watson statistic between 0 and 1. ◦

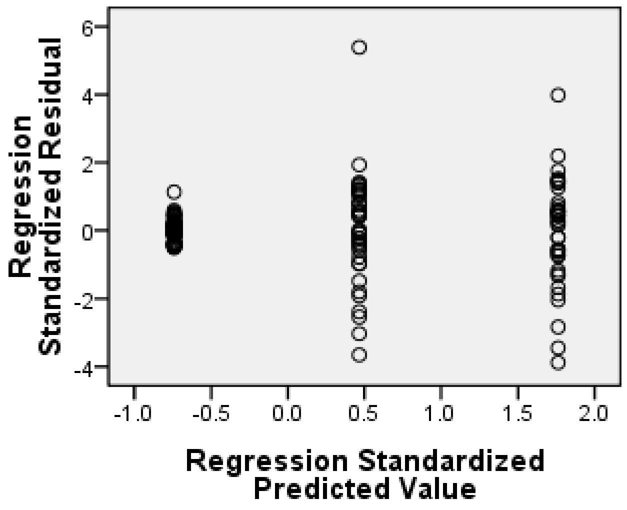


Figure B.91: Residuals against predicted values plot for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals indicates heteroscedasticity. ◦

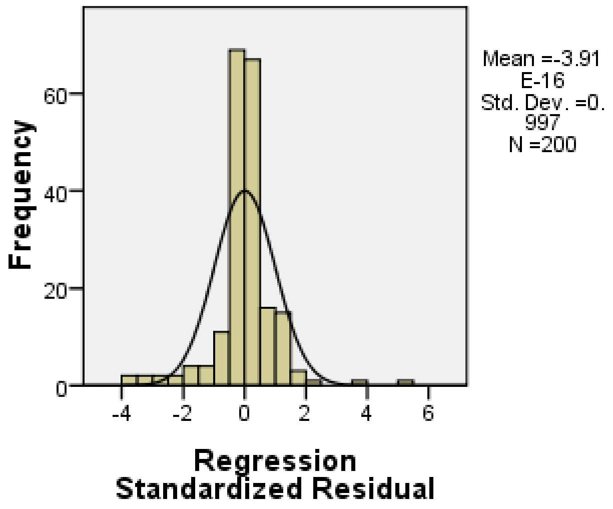


Figure B.92: Histogram of the standardized residuals with normal curve for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The curve is approximated to some extent. ◦

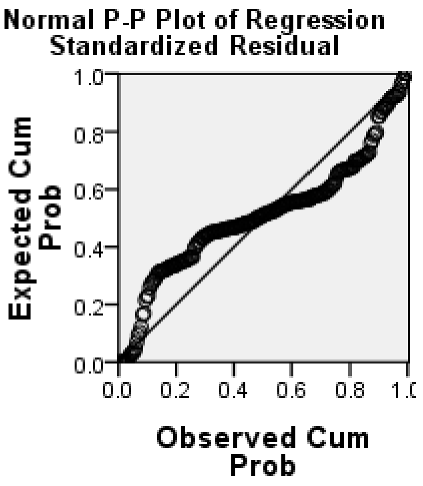


Figure B.93: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The points approximate the straight line to some extent, approximating a normal distribution. ∘

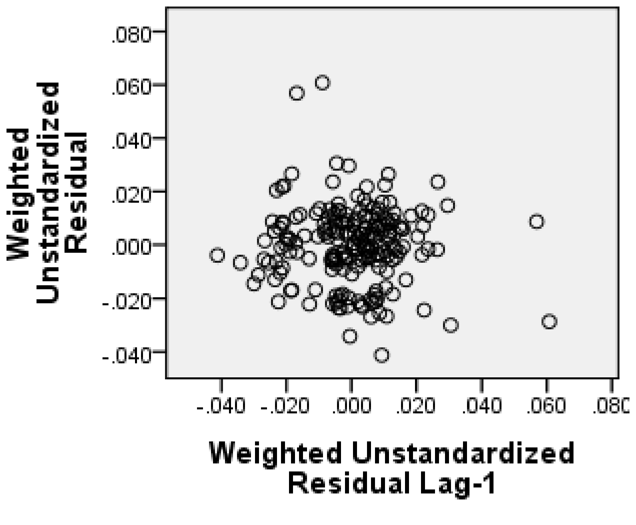


Figure B.94: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The plot indicates hardly any autocorrelation, which is in line with the Durbin-Watson statistic around 2. ∘

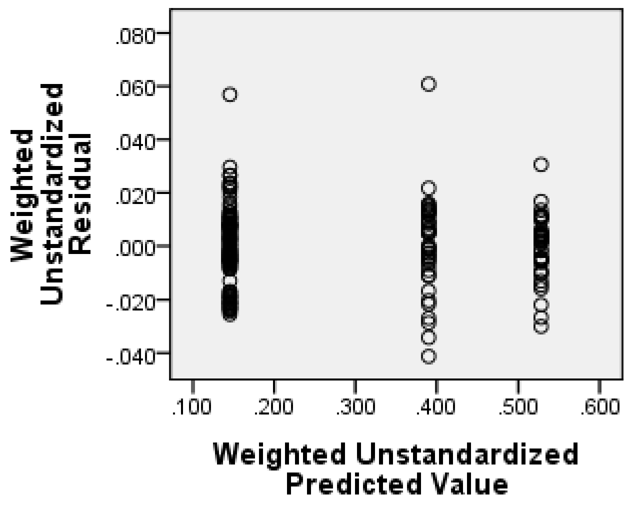


Figure B.95: Residuals-predicted plot for the memory space usage weighted model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals is almost homoscedastic. ∘

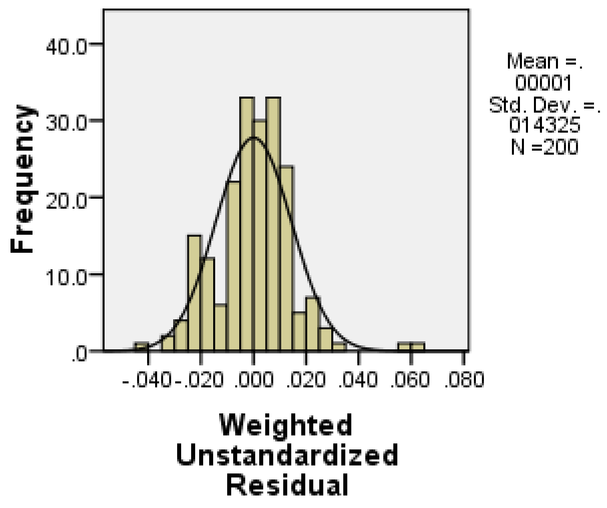


Figure B.96: Histogram of the standardized residuals with normal curve for the memory space usage weighted model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The curve is approximated well. ∘

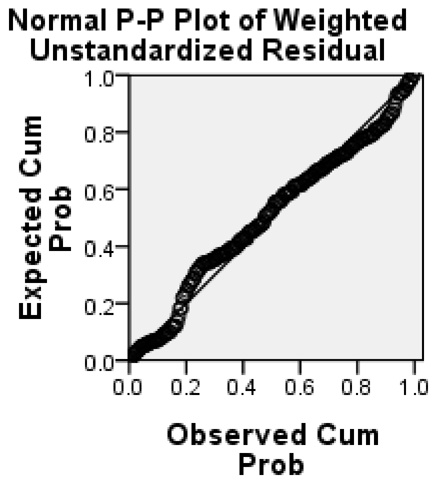


Figure B.97: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage weighted model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The points approximate the straight line, indicating residuals approximate the normal distribution. ◦

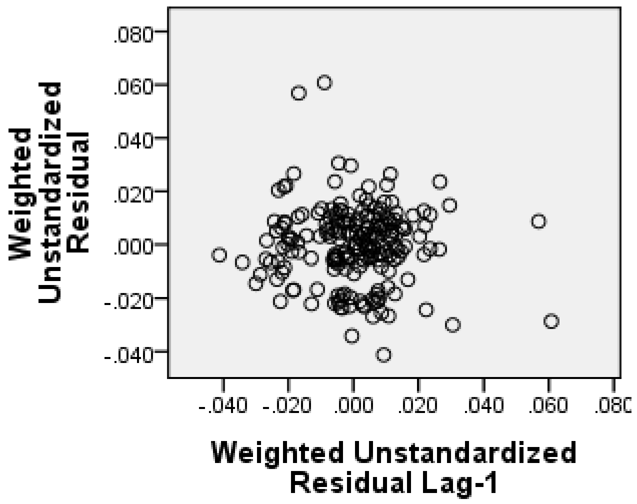


Figure B.98: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage weighted model of the incremental approach with $\epsilon = 0.01$ and $\delta = 0.0001$. The linear relationship indicates hardly any positive autocorrelation. ◦

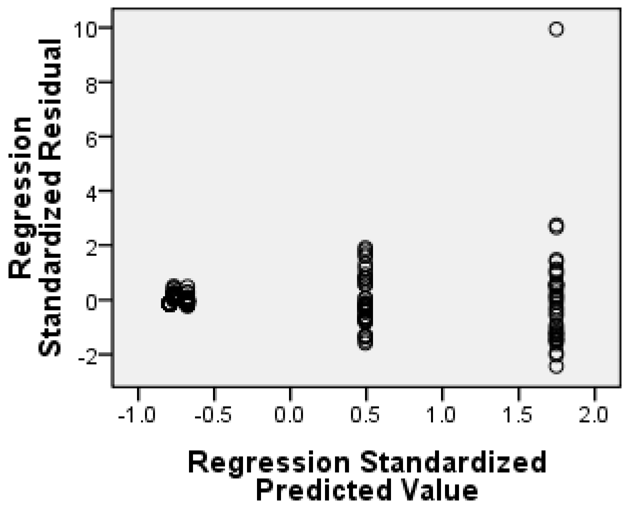


Figure B.99: Residuals against predicted values plot for the processing time model of the incremental approach using expiration lists. The spread of the residuals shows a funnel pattern, indicating heteroscedasticity. ○

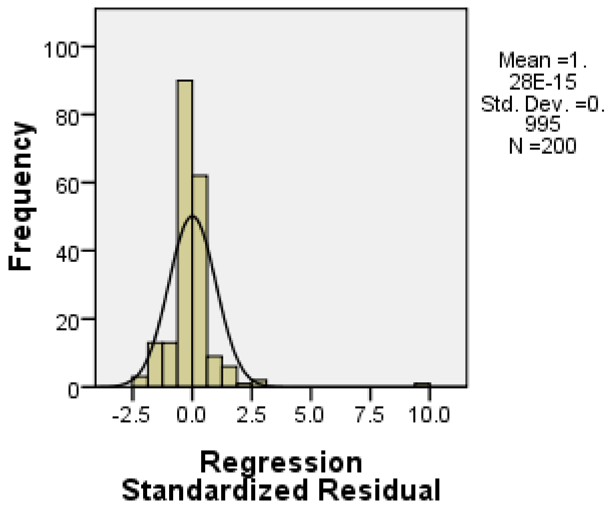


Figure B.100: Histogram of the standardized residuals with normal curve for the processing time model of the incremental approach using expiration lists. The curve is approximated. ○

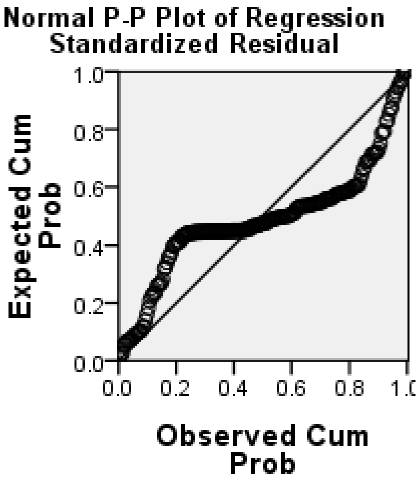


Figure B.101: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the incremental approach using expiration lists. The points are off the straight line, indicating deviation from the normal distribution. ○

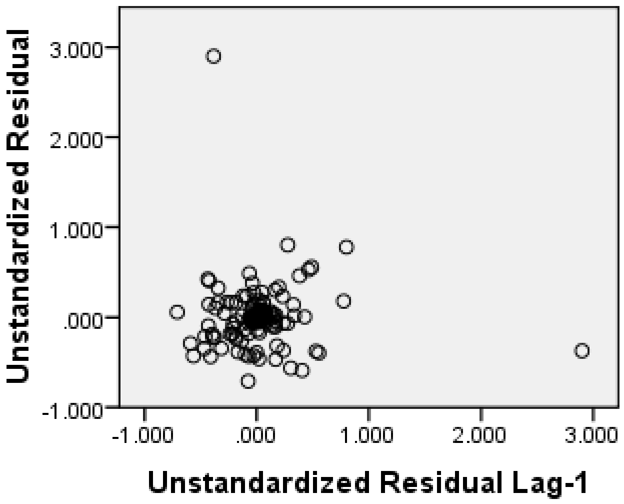


Figure B.102: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the incremental approach using expiration lists. The plot indicates some positive autocorrelation, which is in line with the Durbin-Watson statistic towards 1. ○

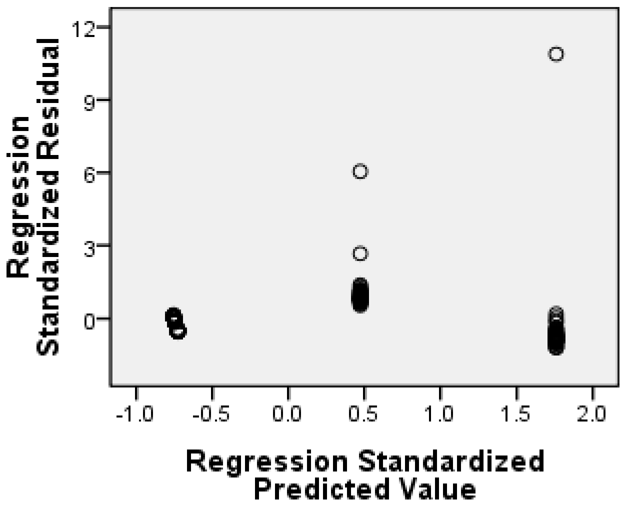


Figure B.103: Residuals against predicted values plot for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The spread of the residuals shows slight heteroscedasticity (apart from some outliers). ∘

B.4 Memory Space Usage: Quantiles

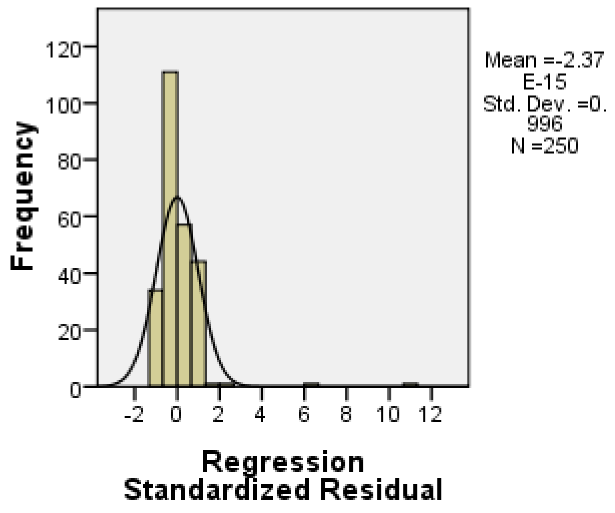


Figure B.104: Histogram of the standardized residuals with normal curve for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The curve is approximated slightly. ○

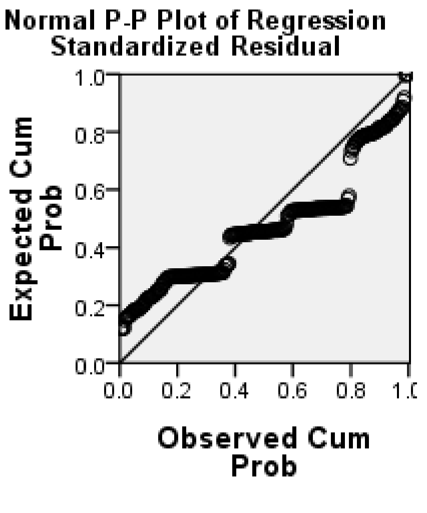


Figure B.105: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The points are off the straight line, indicating deviation from the normal distribution. ○

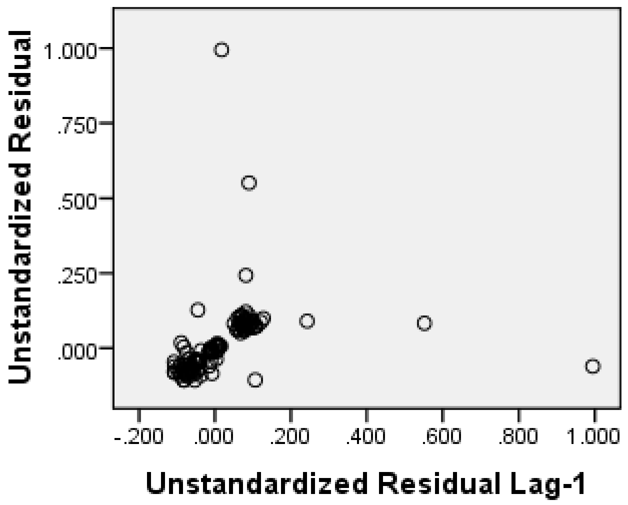


Figure B.106: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The plot indicates positive autocorrelation, which is in line with the Durbin-Watson statistic towards 1. $\circ$

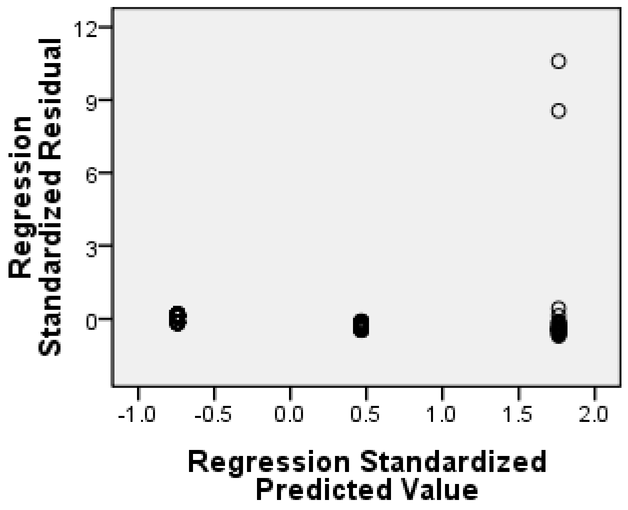


Figure B.107: Residuals against predicted values plot for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The spread of the residuals shows slight heteroscedasticity (apart from some outliers). $\circ$

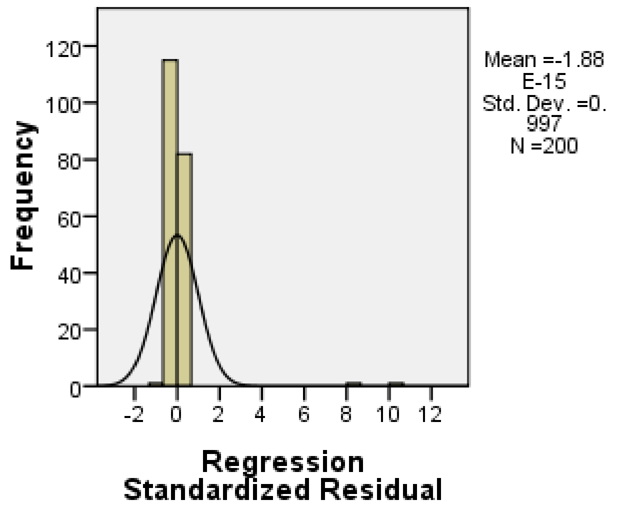


Figure B.108: Histogram of the standardized residuals with normal curve for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The curve is not approximated well. ∘

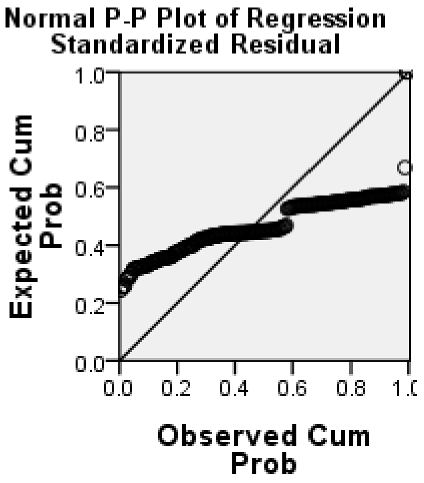


Figure B.109: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The points are off the straight line, indicating clear deviation from the normal distribution. ∘

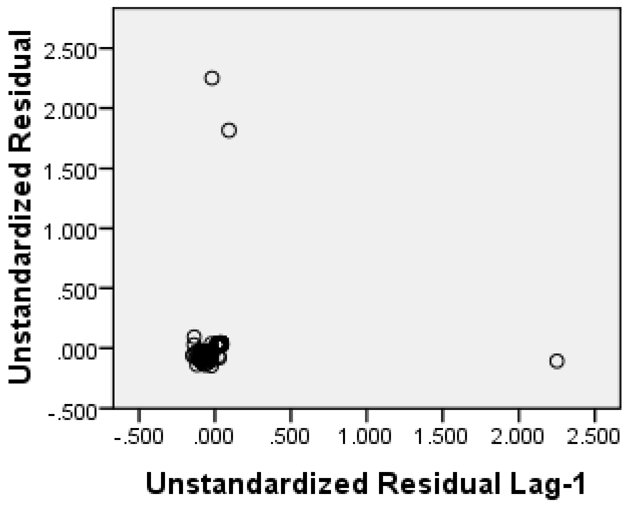


Figure B.110: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the processing time model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The plot indicates some positive autocorrelation, which is in line with the Durbin-Watson statistic between 1 and 2. ◦

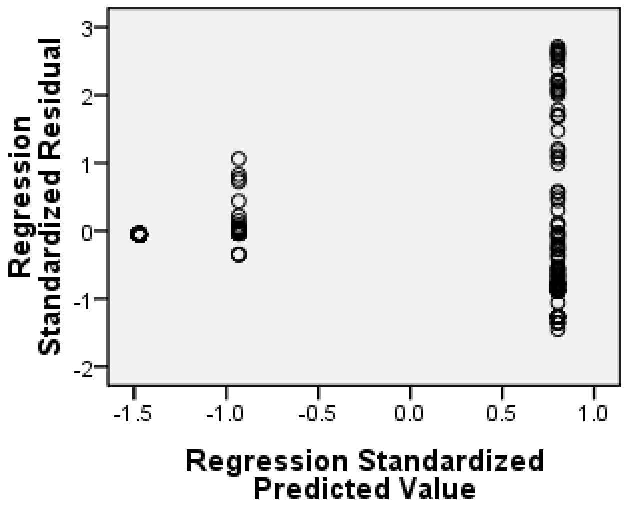


Figure B.111: Residuals-predicted plot for the memory space usage model of the instantaneous approach. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ◦

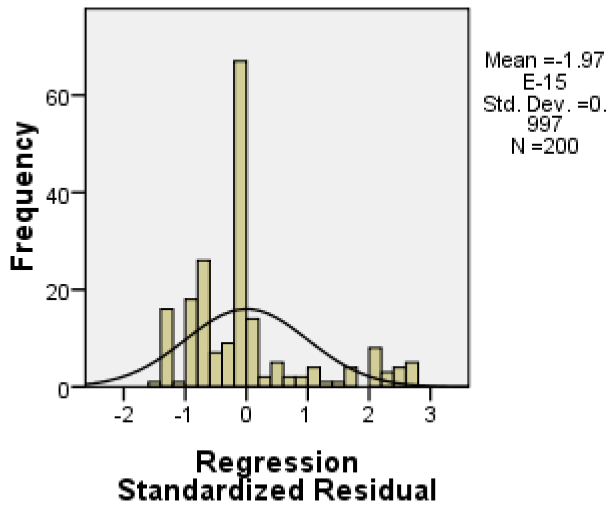


Figure B.112: Histogram of the standardized residuals with normal curve for the memory space usage model of the instantaneous approach. The curve is not approximated well. ◦

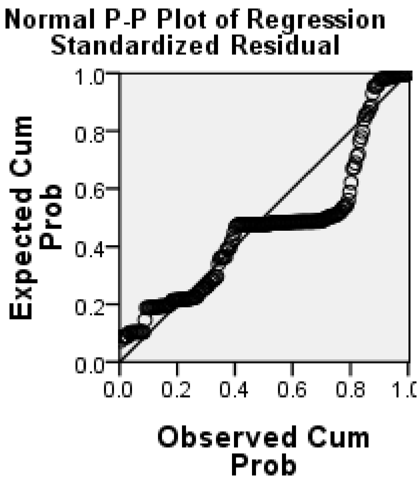


Figure B.113: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the instantaneous approach. The points are clearly off the straight line, indicating deviation from the normal distribution. ◦

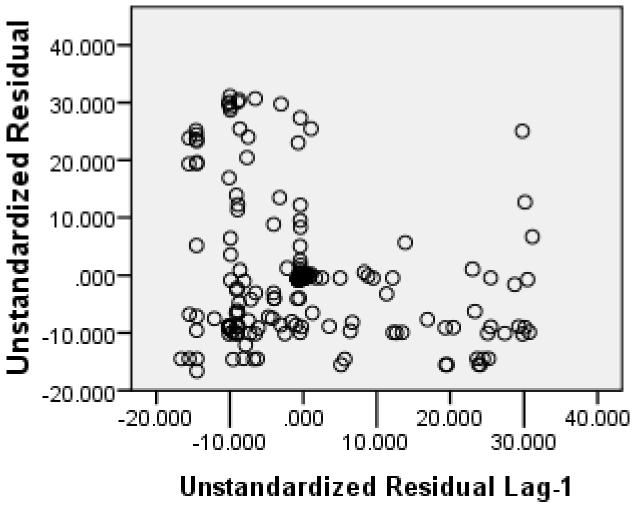


Figure B.114: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the instantaneous approach. The linear relationship indicates strong positive autocorrelation, which is in line with the Durbin-Watson statistic between 0 to 1. ∘

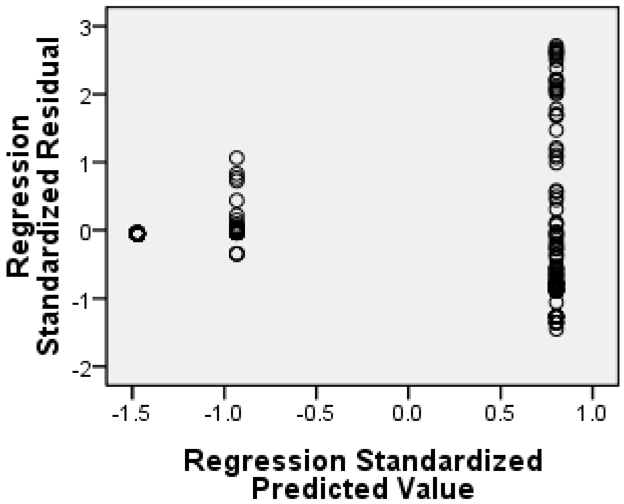


Figure B.115: Residuals-predicted plot for the memory space usage model of the instantaneous approach. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. However, it improves over the unweighted model. ∘

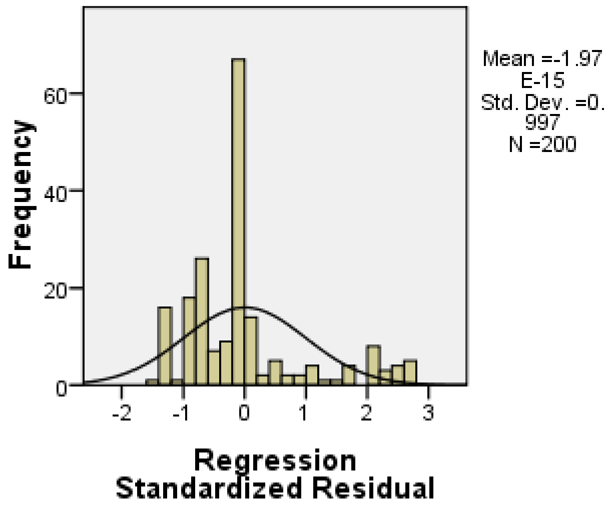


Figure B.116: Histogram of the standardized residuals with normal curve for the memory space usage model of the instantaneous approach. The curve is slightly approximated; better than for unweighted model. ○

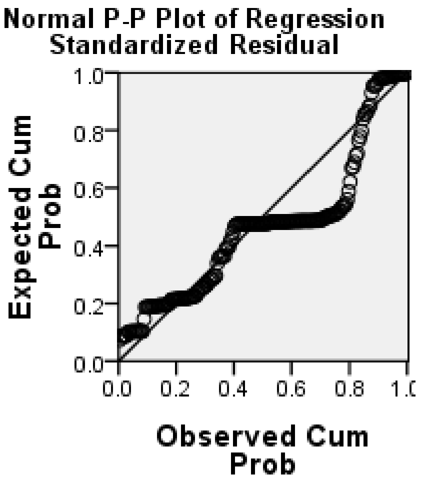


Figure B.117: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the instantaneous approach. The points slightly approximate the straight line, indicating some deviation from the normal distribution. ○

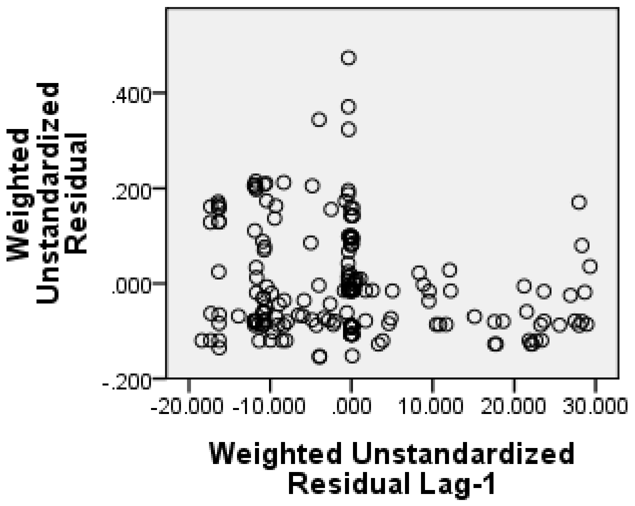


Figure B.118: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the instantaneous approach. The linear relationship indicates strong positive autocorrelation, which is in line with the Durbin-Watson statistic between 0 to 1. ○

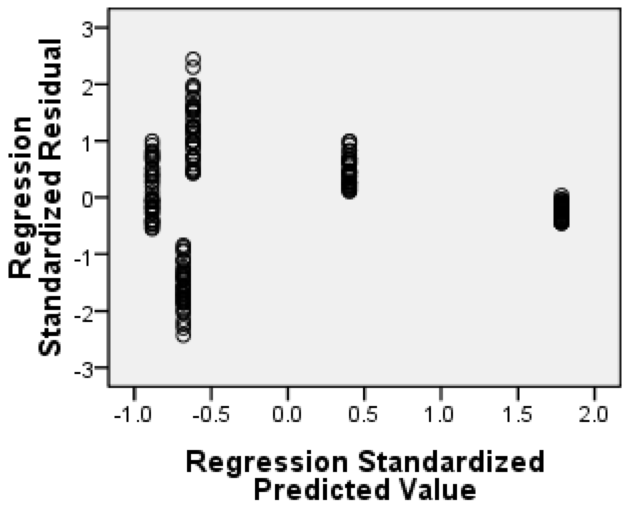


Figure B.119: Residuals-predicted plot for the memory space usage model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ◦

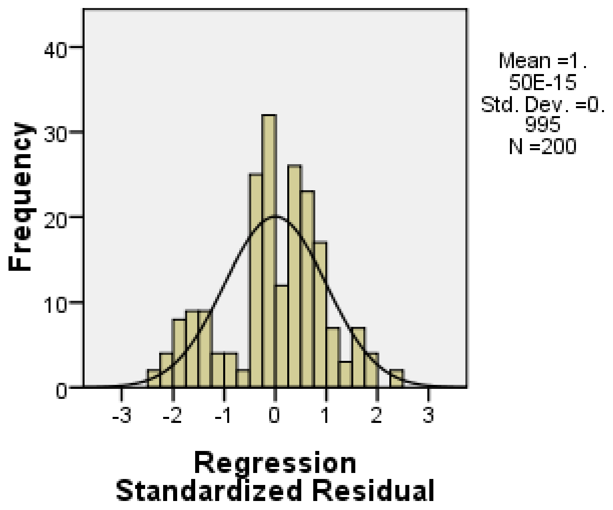


Figure B.120: Histogram of the standardized residuals with normal curve for the memory space usage model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The curve is approximated well. ◦

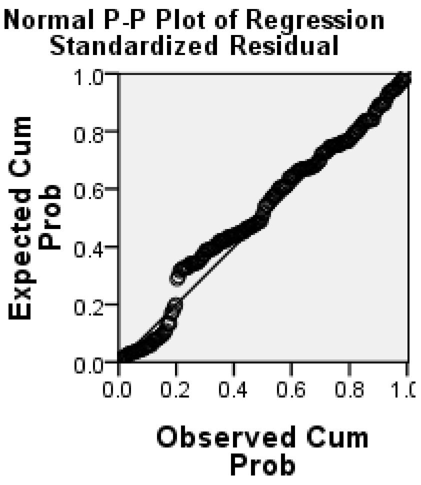


Figure B.121: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The points approximate the straight line, indicating residuals approximate the normal distribution. ◦

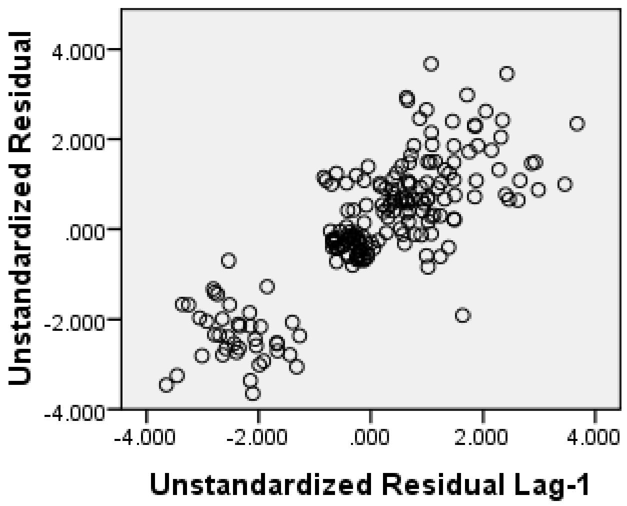


Figure B.122: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the incremental approach with $\epsilon = 0.05$ and $\delta = 0.0001$. The linear relationship indicates positive autocorrelation, which is in line with the Durbin-Watson statistic between 0 to 1. ◦

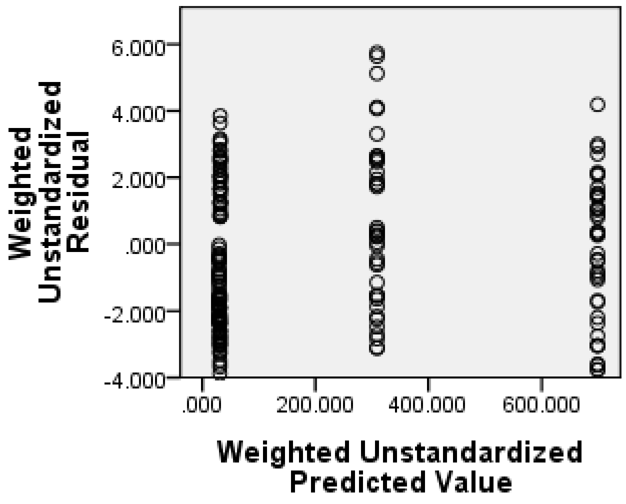


Figure B.123: Residuals-predicted plot for the memory space usage weighted model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The spread of the residuals is almost homoscedastic. ◦

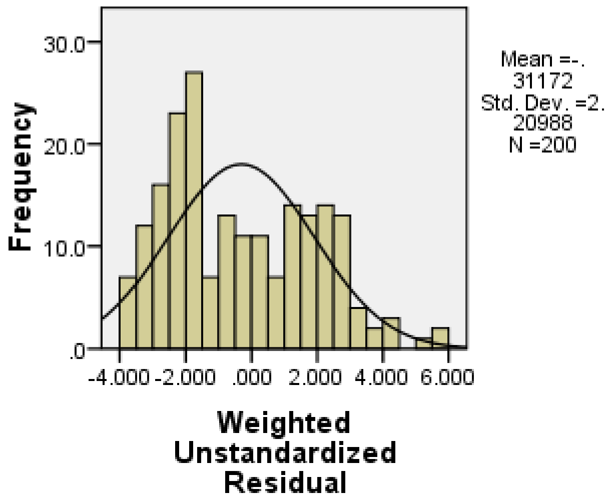


Figure B.124: Histogram of the standardized residuals with normal curve for the memory space usage weighted model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The curve is approximated well. ◦

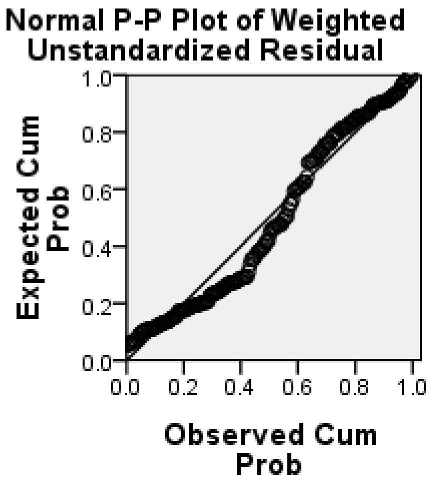


Figure B.125: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage weighted model of the incremental approach using biased reservoir sampling with $\epsilon = 0.05$ and $\delta = 0.0001$. The points approximate the straight line, indicating residuals approximate the normal distribution. ◦

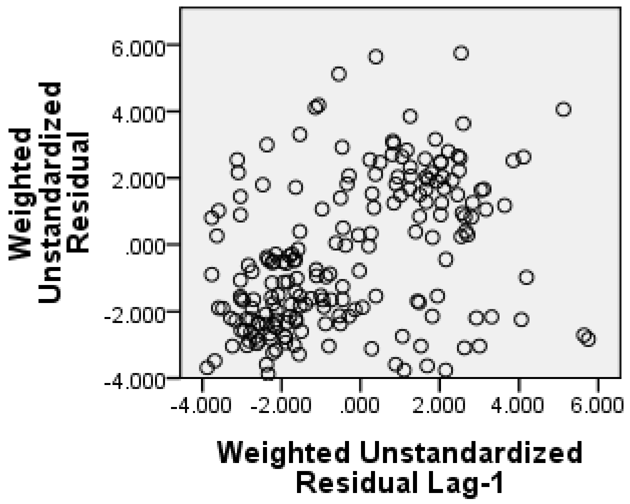


Figure B.126: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage weighted model of the incremental approach with $\epsilon = 0.05$ and $\delta = 0.0001$. The linear relationship indicates some positive autocorrelation. ◦

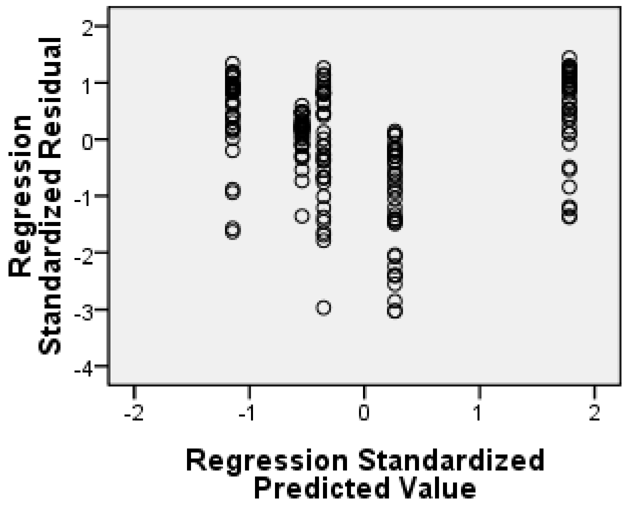


Figure B.127: Residuals-predicted plot for the memory space usage model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The spread of the residuals seems almost homoscedastic. ∘

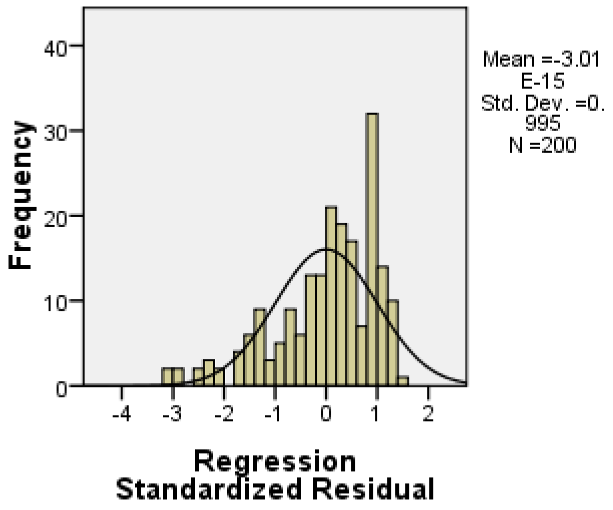


Figure B.128: Histogram of the standardized residuals with normal curve for the memory space usage model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The curve is approximated well. ∘

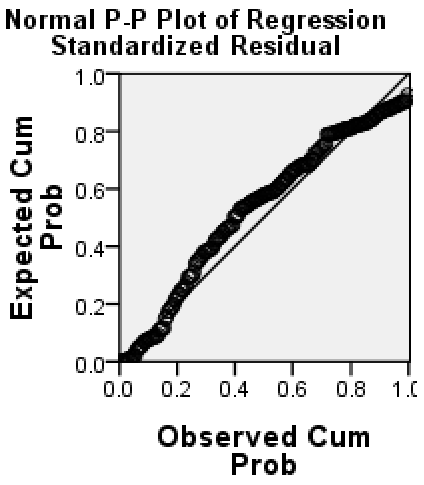


Figure B.129: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the incremental approach using biased reservoir sampling with $\epsilon = 0.01$ and $\delta = 0.0001$. The points approximate the straight line, indicating residuals approximate the normal distribution. ∘

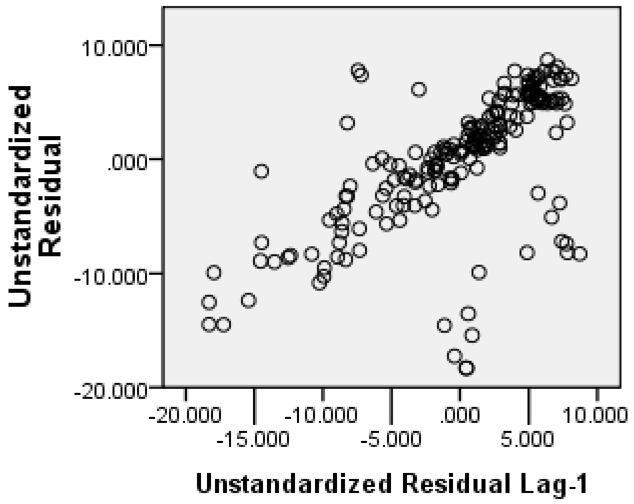


Figure B.130: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the incremental approach with $\epsilon = 0.01$ and $\delta = 0.0001$. The linear relationship indicates positive autocorrelation, which is in line with the Durbin-Watson statistic between 0 to 1. ∘

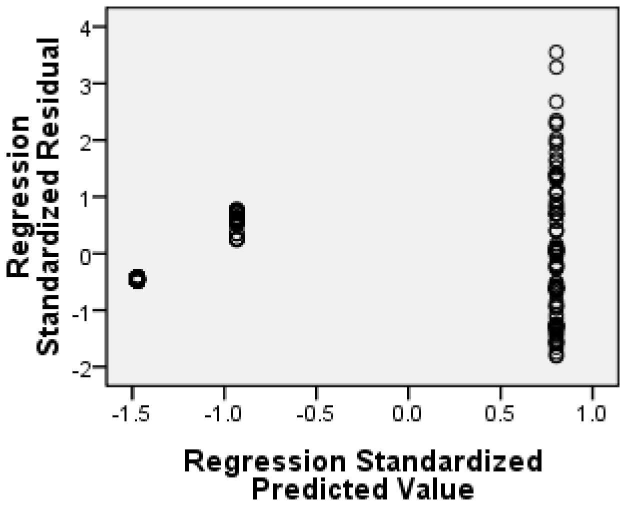


Figure B.131: Residuals-predicted plot for the memory space usage model of the incremental approach using expiration lists. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. The heteroscedasticity is very strong. ◦

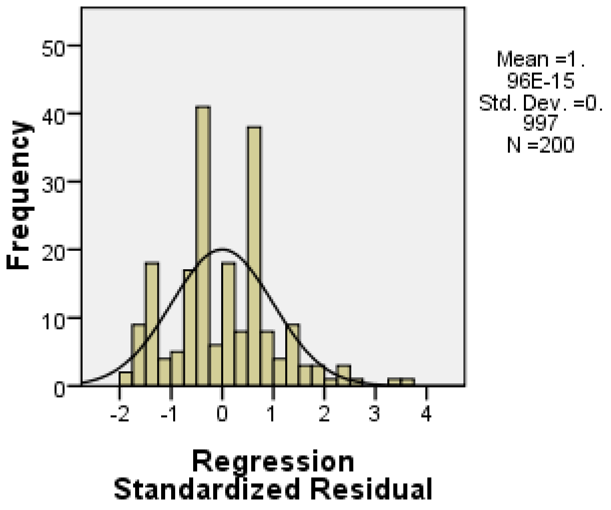


Figure B.132: Histogram of the standardized residuals with normal curve for the memory space usage model of the incremental approach using expiration lists. The curve is slightly approximated. ◦

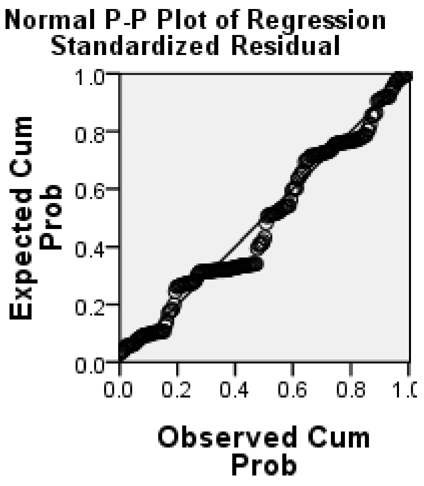


Figure B.133: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the incremental approach using expiration lists. The points approximate the straight line, indicating some deviation from the normal distribution. ○

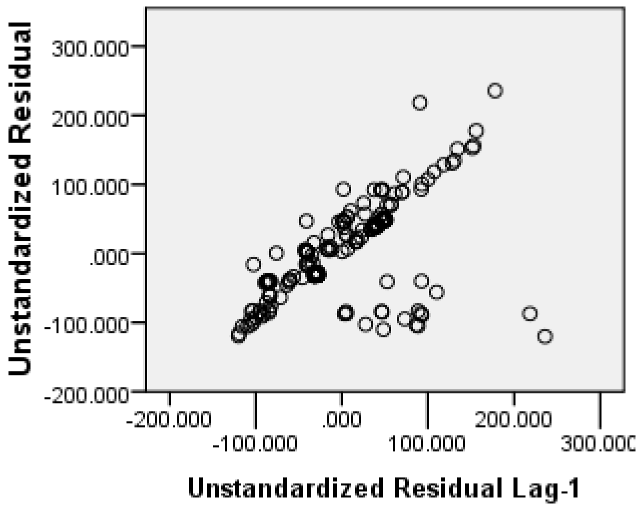


Figure B.134: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the incremental approach using expiration lists. The linear relationship indicates very strong positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ○

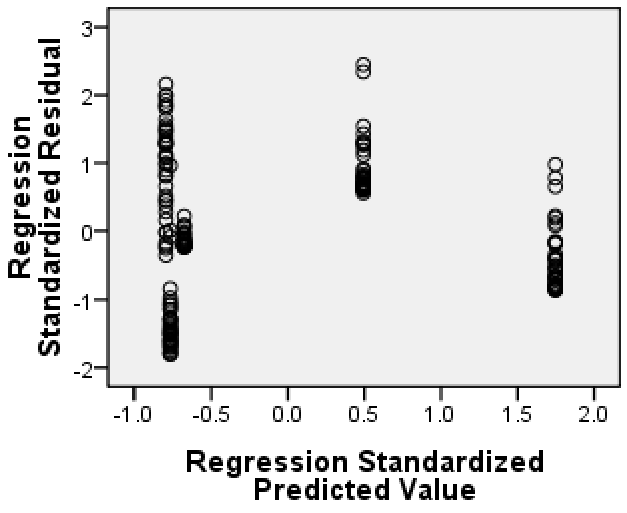


Figure B.135: Residuals-predicted plot for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The spread of the residuals varies non-uniformly with the predicted values, indicating heteroscedasticity. ∘

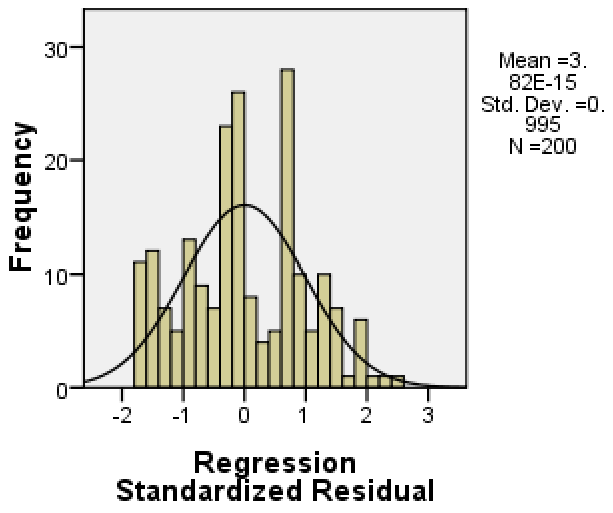


Figure B.136: Histogram of the standardized residuals with normal curve for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The curve is slightly approximated. ∘

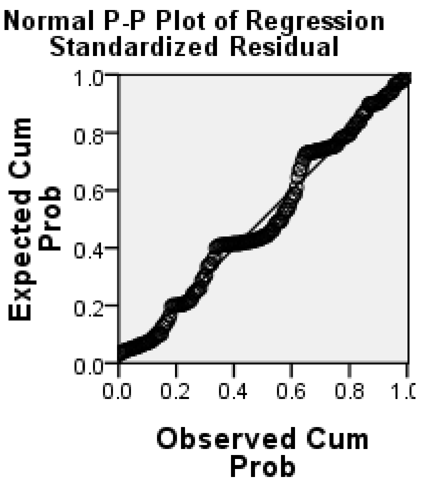


Figure B.137: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The points approximate the straight line, indicating some deviation from the normal distribution. ◦

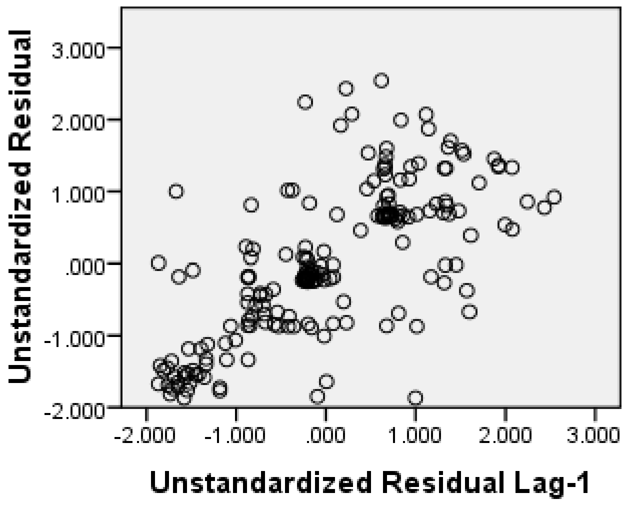


Figure B.138: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.05$. The linear relationship indicates strong positive autocorrelation, which is in line with the Durbin-Watson statistic towards 0. ◦

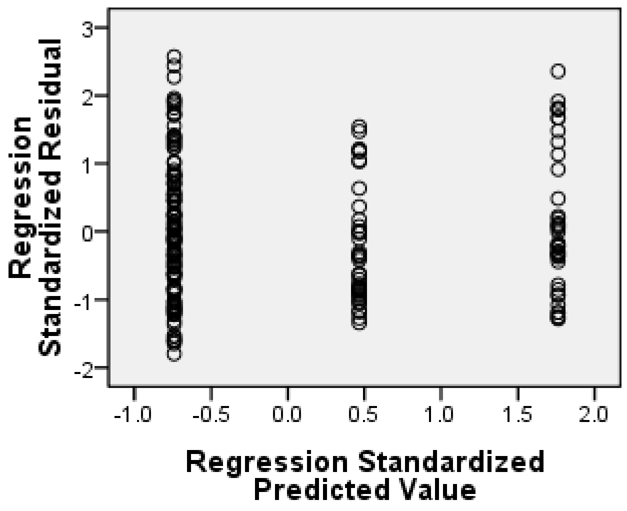


Figure B.139: Residuals-predicted plot for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The spread of the residuals shows hardly any heteroscedasticity. ○

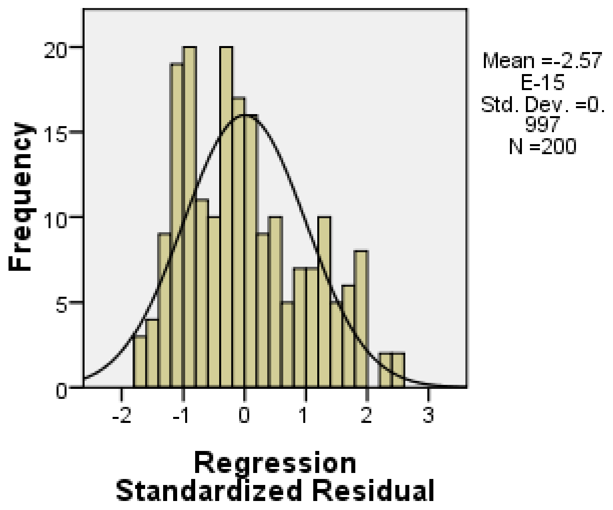


Figure B.140: Histogram of the standardized residuals with normal curve for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The curve is approximated. ○

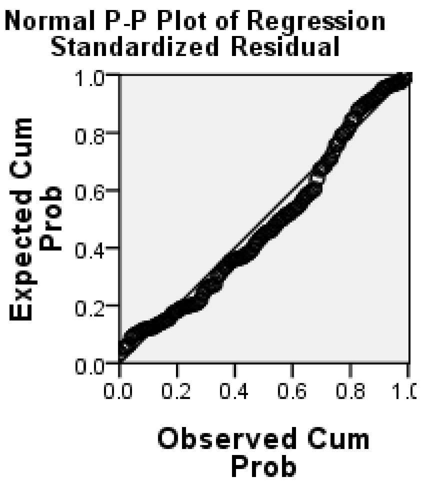


Figure B.141: Normal P-P plot of the (cumulative) normal distribution against the (cumulative) residuals distribution for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The points do approximate the straight line, indicating slight deviation from the normal distribution. ◦

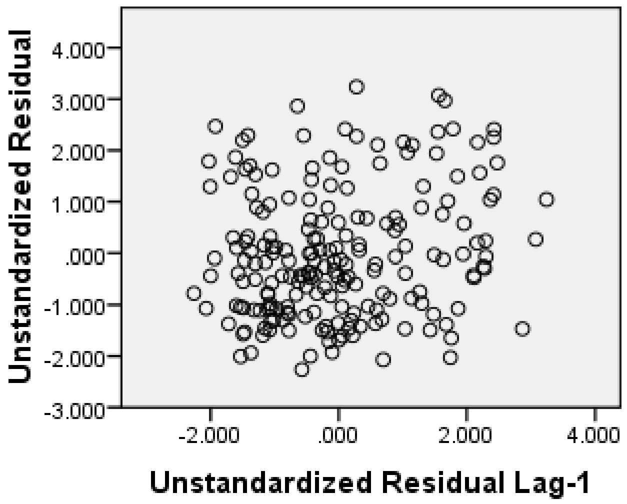


Figure B.142: Lag plot of unstandardized residuals e_i against lag 1 unstandardized residuals e_{i-1} for the memory space usage model of the incremental approach using Arasu-Manku fixed sketches with $\epsilon = 0.01$. The linear relationship indicates slight positive autocorrelation, which is in line with the Durbin-Watson statistic towards 2. ◦