FAST ALGORITHMS FOR COMPUTING ODD MOMENTS IN STATISTICAL ANALYSIS OF PRIVACY-RELATED INTERVAL DATA

JORGE IVAN VARGAS

Department of Computer Science

APPROVED:

Vladik Kreinovich, Chair, Ph.D.

Luc Longpré, Ph.D.

Luis Valdez-Sanchez, Ph.D.

Pablo Arenaz, Ph.D.
Dean of the Graduate School

to my

MOTHER and FATHER

with love

FAST ALGORITHMS FOR COMPUTING ODD MOMENTS IN STATISTICAL ANALYSIS OF PRIVACY-RELATED INTERVAL DATA

by

JORGE IVAN VARGAS

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

July 2007

Acknowledgments

I would like to take advantage of this space to thank all people that have supported me throughout my time as a student: faculty, classmates, and family.

Regarding academic environment, I want to thank Dr. Vladik Kreinovich for all his time, dedication, enthusiasm, and patience. I do not have enough words to express my gratitude. I can honestly tell, he is one of the most student-oriented professors I have met in years as a student. Without him, the journey would had been a lot more difficult.

I want to thank as well Dr. Luis Valdez-Sanchez from the Math Department. Even though he joined my committee in an advanced stage of my thesis, he proved to be a great mentor and friend. I wish I would had met him before.

To Dr. Longpré, my deepest appreciation. He had always been there to support me and willing to help.

I must admit, college years are stressful and often painful. Fortunately, I was not alone in this journey. I had the chance to meet other students, many of whom are now professionals working for big companies, others decided to continue studying, but all of them are still great friends of mine. There are many moments that I have shared with many of them that I will never forget.

And overall, thanks to all the faculty and staff members from the Computer Science Department. I feel very honored because I had the opportunity to be a student of great professors such as Dr. Pat Teller, Dr. Steve Roach, Dr. Nigel Ward, Dr. Yoonsik Cheon. I do not have enough room to thank all my professors who had guided me to succeed both academically and personally.

I want to thank also, the staff from The University of Texas System Louis Stokes Alliance for Minority Participation, especially Dr. Ben Flores, Ms. Ariana Aciero, and my fellow students which were also supported by this program. I really feel very proud and honored for have been selected to be part of the 2005-2007 UTEP Bridge-to-the-Doctorate

generation. I am very happy because I had the opportunity to meet brilliant people – my fellow students – but more than that, great friends.

Regarding my family, I am very lucky to have such a big and great family. I could type twice as many pages as there are in this thesis expressing my feeling and gratitude to them, but I will keep it short.

First of all, thanks to my parents. During all my years as a college students, they were always there to support me. I cannot really tell how I feel about them. I will never forget the advice given by my mother, all those hours we spent together talking late night, even when she was very tired, but many time that was the only free time I had. To my father, thanks for all your support and understanding. As well as my mother, thanks for the time we spent together in family.

Finally, thanks to my wife, Vicky, for all her patience and understanding these last years. I know it is not easy to be living with a college student, but somehow she managed to make our life much easier. She and our children, Fernando and Vanessa, are the source of my inspiration. I love you, Vicky.

NOTE: This thesis was submitted to my Supervising Committee on the July 15, 2007.

Abstract

In many practical situations, we perform statistical analysis of data about human objects. For example, the main purpose of a census is to enable us to get statistical information about the population: how fast does it grow, what are the trends in population and income change, etc. These trends can be found if we find the mean values of different characteristics, degrees of deviation from these mean values (e.g., variance and higher moments), correlations between different characteristics, etc. Similarly, it is important to find the correlation between the effectiveness of a certain treatment and age, gender, etc., of patients.

In all these situations, it is important to maximally preserve the privacy of the people whose data is being processed. One way to preserve this privacy is not to store the actual values of the corresponding quantities, but only to store ranges (intervals). For example, instead of the actual value of the age, we store the range (0 to 10, 10 to 20, etc.). This interval representation solves the privacy problem, but a new problem appears: how to perform statistical analysis of this interval data? Different values x_i from the given ranges $[\underline{x}_i, \overline{x}_i]$, in general, lead to different values of the desired statistical characteristic $C(x_1, \ldots, x_n)$. It is therefore necessary to find the range $\mathbf{C} = [\underline{C}, \overline{C}]$ of possible values of the desired characteristic $C(x_1, \ldots, x_n)$ when $x_i \in [\underline{x}_i, \overline{x}_i]$.

Several researchers have developed efficient algorithms for statistical processing under such privacy-related interval uncertainty. Specifically, efficient algorithms have been designed for computing the range $\mathbf{C}$ for mean, variance, central moments M_{2p}, and for the third central moment M_3.

Higher odd moments are also important in describing the shape of the empirical distributions. For computing these moments under privacy-related interval uncertainty, no efficient algorithm was previously known. In this thesis, we design a new efficient $\mathcal{O}(n^2)$ algorithm for computing the range for odd central moments under privacy-related interval uncertainty.

Table of Contents

List of Tables

Chapter 1

Interval Approach to Preserving Privacy in Statistical Databases: Introduction to the General Problem

1.1 Need for Statistical Databases

In many practical situations, it is very useful to collect large amounts of data.

For example, from the data that we collect during a census, we can extract a lot of information about health, mortality, employment in different regions – for different age ranges, and for people from different genders and of different ethnic groups. By analyzing this statistics, we can reveal troubling spots and allocate (usually limited) resources so that the help goes first to social groups that need it most.

Similarly, by gathering data about people's health in a large medical database, we can extract a lot of useful information on how the geographic location, age, and gender affect a person's health. Thus, we can make measures which are aimed at improving public health, more focused.

Finally, a large statistical database of purchases can help find out what people are looking for, make shopping easier for customers and at the same time, decrease the stores' expenses related to storing unnecessary items.

1.2 Need for Privacy

Privacy is an important issue in the statistical analysis of human-related data. For example, to check whether in a certain geographic area, there is a gender-based discrimination, we can use the census data to check, e.g., whether for all people from this area who have the same level of education, there is a correlation between salary and gender. One can think of numerous possible questions of this type related to different sociological, political, medical, economic, and other questions. From this viewpoint, it is desirable to give researchers *ability to perform* whatever *statistical analysis* of this data that is reasonable for their specific research.

On the other hand, we do not want to give them direct access to the raw census data, because a large part of the census data is *confidential*. For example, for most people (those who work in private sector) salary information is confidential. Suppose that a corporation is deciding where to built a new plant and has not yet decided between two possible areas. This corporation would benefit from knowing the average salary of people of needed education level in these two areas, because this information would help them estimate how much it will cost to bring local people on board. However, since salary information is confidential, the company should not be able to know the exact salaries of different potential workers.

The need for privacy is also extremely important for *medical* experiments, where we should be able to make statistical conclusions about, e.g., the efficiency of a new medicine without disclosing any potentially embarrassing details from the individual medical records.

Such databases in which the outside users cannot access individual records but can solicit statistical information are often called *statistical databases*.

1.3 Maintaining Privacy is Not Easy

Maintaining privacy in statistical databases is not easy. Clerks who set up policies on access to statistical databases sometimes erroneously assume that once the records are made anonymous, we have achieved perfect privacy. Alas, the situation is not so easy: even when we keep all the records anonymous, we can still extract confidential information by asking appropriate questions. For example, suppose that we are interested in the salary of Dr. X who works for a local company. Dr. X's mailing address can be usually taken from the phone book; from the company's webpage, we can often get his photo and thus find out his race and approximate age. Then, to determine Dr. X's salary, all we need is to ask what is the average salary of all people with a Ph.D. of certain age brackets who live in a small geographical area around his actual home address – if the area is small enough, then Dr. X will be the only person falling under all these categories.

Even if we only allow statistical information about salaries $s_1, \ldots, s_q$ when there are at least a certain amount n_0 people within a requested range, we will still be able to reconstruct the exact salaries of all these people. Indeed, for example, we can ask for the number and average salary of all the people who live on Robinson street at houses 1 through 1001, and then we can ask the same question about all the people who live in houses from 1 to 1002. By comparing the two numbers, we get the average salary of the family living at 1002 Robinson – in other words, we gain the private information that we tried to protect. Several other hacking techniques are described, e.g., in [23].

In general, we can ask for the average $\dfrac{s_1 + \ldots + s_q}{q}$, and for several moments of salary (variance, third moment, etc): if we know the values v_j at least q different functions $f_j(s_1, \ldots, s_q)$ of s_i, then we can, in general, reconstruct all these values from the corresponding system of q equations with q unknowns: $f_1(s_1 \ldots, s_q) = v_1$, $\ldots$, $f_q(s_1, \ldots, s_q) = v_q$.

At first glance, moments are natural and legitimate statistical characteristics, so researchers would be able to request them, but on the other hand, we do not want them to be able to extract the exact up-to-cent salaries of all the folks leaving in a certain

geographical area.

What restriction should we impose on possible statistical queries that would guarantee privacy but restrict research in the least possible way?

1.4 Maintaining Privacy: What Is Known

These are anecdotal example of poorly designed privacy and security, but, as we have mentioned, the problem is indeed difficult: many seemingly well-designed privacy schemes later turn out to have unexpected privacy and security problem.

For different aspects of the problem of privacy in statistical databases, and for different proposed solution to this problem and their drawbacks, the readers are referred to [2, 7, 8, 9, 10, 11, 19, 20, 25, 27, 28, 29, 33, 34, 35, 37]; an extended bibliography of pre-1980s papers appears in Chapter 6 of [8].

That the privacy problem is really difficult was confirmed by the fact that several formalizations of this general privacy problem turned out to be, in their *general* formulations, NP-hard [8].

1.5 Interval Approach to Privacy Protection

A sure-proof way to avoid these privacy violations is to store ranges (intervals) of values instead of the actual values. For example, instead of keeping the exact age, we only record whether the age is between 0 and 10, 10 and 20, 20 and 30, etc.

In this case, no matter what statistics we allow, the worst that can happen is that the corresponding ranges will be disclosed. However, in this situation, we do not disclose the original exact values – since these values are not stored in the database in the first place.

1.6 Related Challenges of Computational Statistics

The idea of storing intervals solves the privacy problem, but it leads to a computational challenge.

Indeed, suppose that we are interested in the value of a statistical characteristic $C(x_1, \ldots, x_n)$ such as population mean

$$E = \frac{x_1 + \ldots + x_n}{n},$$

(biased) population variance

$$V = \frac{(x_1 - E)^2 + \ldots + (x_n - E)^2}{n},$$

covariance, correlation, etc.

Traditional statistical algorithms for computing these characteristics assume that we know the exact values of the samples x_i, y_i, etc. However, in our case, we do not know these actual values, we only know the *intervals* $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ of possible values of these characteristics. Since we do not know the actual values x_i, we cannot compute the exact range of the characteristic C, we can only find the *range* of this characteristic:

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{C(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

The function $C(x_1, \ldots, x_n)$ is usually continuous; since the domain

$$\mathbf{x}_1 \times \ldots \times \mathbf{x}_n \stackrel{\text{def}}{=} \{(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

is a bounded closed connected set, the range $C(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is an interval.

So, the challenge is: given the characteristic $C(x_1, \ldots, x_n)$ and the intervals $\mathbf{x}_i$, we must compute the corresponding interval range.

1.7 Specific Challenge that We Handle in This Thesis

As we will show in the following chapters, the problem of computing statistical characteristics under privacy-related interval uncertainty has already been studied. For many

practically important statistical characteristics, there exist algorithms for computing them under privacy-related interval uncertainty.

For example, there exist algorithms for computing the mean, the standard deviation, etc. There was only one case when such an algorithm was not known: the general case of odd central moments. These moments are very important. For example, the third central moment (skewness) is used to check whether the distribution is symmetric; higher odd moments provide an even better description of the possible asymmetry. For odd moments, only an algorithm for skewness was known, but no algorithms were known for computing 5-th, 7-th and other higher order odd central moments.

In this thesis, we close this gap and produce an efficient $\mathcal{O}(n^2)$ algorithm which computes the range of odd moments under privacy-related interval uncertainty. This result was mentioned in a publication [22].

1.8 Outline of the Thesis

In Chapters 2, we overview the general interval computations techniques. In Chapter 3, we describe the known results about the computational complexity of computing statistical characteristics under interval uncertainty. In Chapter 4, we describe, in detail, known algorithms for computing statistical characteristics under privacy-related interval uncertainty, and the corresponding results.

Chapter 5 contains the main result of this thesis: a new algorithm for computing odd moments under privacy-related interval uncertainty, with a proof that this algorithm is indeed correct and efficient. Chapter 6 contains conclusions and ideas for future work.

The code of the program implementing the new algorithm and the instructions for using this program are given in the Appendix A.

Chapter 2

Computations under Interval Uncertainty: What Is Known

2.1 Privacy-Related Interval Uncertainty: Reminder

As we have mentioned in Chapter 1, to preserve privacy in a statistical databases, instead of storing the actual values x_i of the corresponding quantities, we only store intervals $[\underline{x}_i, \overline{x}_i]$ of possible values of these quantities.

Usually, we have several thresholds $0 = t_0 < t_1 < \ldots < t_q$ that divide the set of possible values into intervals $[0, t_1]$, $[t_1, t_2]$, $\ldots$, $[t_{q-1}, t_q]$. Instead of storing the original value, we only store the interval $[t_i, t_{i+1}]$ which contains this quantity.

For example, instead of storing the actual age of a person, we only store the information on whether this person person is under 10 years of age, from 10 to 20, from 20 to 30, etc. This corresponds to the thresholds $t_0 = 0$, $t_1 = 10$, $t_2 = 20$, etc., and intervals $[t_0, t_1] = [0, 10]$, $[t_1, t_2] = [10, 20]$, $[t_2, t_3] = [20, 30]$, $\ldots$

In general, for privacy-related interval data, every two intervals either coincide or have at most one point in common (if they do have a point in common, this point must be an endpoint of each of the intervals).

Let us illustrate possible situations on three examples. These examples cover the case of the ages of people from the same household. Let us assume that in this household, we have 3 brothers of ages 14, 16, and 22, and their mother of age 41, and their father of age 45. The corresponding age intervals are $[10, 20]$, $[10, 20]$, $[20, 30]$, $[40, 50]$, and $[40, 50]$.

- As a first example, let us take the two youngest brothers of ages 14 and 16. For both brothers, the corresponding age intervals are $[10, 20]$. These intervals coincide.
- As a second example, we can take the youngest brother whose age is 14 and the eldest brother whose age is 22. For the first brother, the corresponding age interval is $[10, 20]$. For the second brother, the corresponding age interval is $[20, 30]$. These two intervals have a common point 20 which is an endpoint of both intervals: it is the upper endpoint of the interval $[10, 20]$ and the lower endpoint of the interval $[20, 30]$.
- As a third example, we compare the age of the youngest brother of age 14 and the age of his mother (41). The corresponding intervals $[10, 20]$ and $[40, 50]$ do not have any points in common.

In more precise terms, we say that for every two intervals $[a, b]$ and $[c, d]$, if $(a, b) \cap (c, d) \neq \emptyset$, then $[a, b] = [c, d]$.

We will use this property as a formal definition of privacy-related interval uncertainty.

2.2 Related Challenges of Computational Statistics: Reminder

Indeed, suppose that we are interested in the value of a statistical characteristic $C(x_1, \ldots, x_n)$ such as population mean

$$E = \frac{x_1 + \ldots + x_n}{n},$$

(biased) population variance

$$V = \frac{(x_1 - E)^2 + \ldots + (x_n - E)^2}{n},$$

covariance, correlation.

Traditional statistical algorithms for computing these characteristics assume that we know the exact values of the samples x_i, y_i, etc. However, in our case, we do not know

these actual values, we only know the *intervals* $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ of possible values of these characteristics. Since we do not know the actual values x_i, we cannot compute the exact range of the characteristic C, we can only find the *range* of this characteristic:

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{C(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

So, the challenge is: given the characteristic $C(x_1, \ldots, x_n)$ and the intervals $\mathbf{x}_i$, we must compute the corresponding range.

2.3 The Corresponding Computational Problem is a Particular Case of Interval Computations

While privacy-related applications are reasonably novel, the problem of computing the range of a known function $f(x_1, \ldots, x_n)$ under interval uncertainty $x_i \in \mathbf{x}_i$ is a well-known and well-studied problem in applications, known as a problem of *interval computations*; see, e.g., [14] (see also [26]).

Indeed, in many real-life problems, we are interesting in the values of some quantity y which are difficult or impossible to measure directly; example include the amount of oil in a given well or a distance to a star. To estimate the value of this quantity y, we measure the values of easier-to-measure quantities $x_1, \ldots, x_n$ related to y in a known way $y = f(x_1, \ldots, x_n)$, and then use the measured values $\widetilde{x}_i$ of these quantities to estimate y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

Measurements are never 100% accurate. As a result, the result $\widetilde{x}$ of the measurement is, in general, different from the (unknown) actual value x of the desired quantity. The difference $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$ between the measured and the actual values is usually called a *measurement error*.

The manufacturers of a measuring device usually provide us with an upper bound Δ for the (absolute value of) possible errors, i.e., with a bound Δ for which we guarantee that $|\Delta x| \leq \Delta$. The need for such a bound comes from the very nature of a measurement

process: if no such bound is provided, this means that the difference between the (unknown) actual value x and the observed value $\widetilde{x}$ can be as large as possible.

Since the (absolute value of the) measurement error $\Delta x = \tilde{x} - x$ is bounded by the given bound Δ, we can therefore guarantee that the actual (unknown) value of the desired quantity belongs to the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.

In many practical situations, we not only know the interval $[-\Delta, \Delta]$ of possible values of the measurement error; we also know the probability of different values Δx within this interval [30].

In practice, we can determine the desired probabilities of different values of Δx by comparing the results of measuring with this instrument with the results of measuring the same quantity by a standard (much more accurate) measuring instrument. Since the standard measuring instrument is much more accurate than the one use, the difference between these two measurement results is practically equal to the measurement error; thus, the empirical distribution of this difference is close to the desired probability distribution for measurement error.

There are two cases, however, when this determination is not done:

- First is the case of cutting-edge measurements, e.g., measurements in fundamental science. When a Hubble telescope detects the light from a distant galaxy, there is no "standard" (much more accurate) telescope floating nearby that we can use to calibrate the Hubble: the Hubble telescope is the best we have.

- The second case is the case of measurements on the shop floor. In this case, in principle, every sensor can be thoroughly calibrated, but sensor calibration is so costly – usually costing ten times more than the sensor itself – that manufacturers rarely do it.

In both cases, we have no information about the probabilities of Δx; the only information we have is the upper bound on the measurement error.

In this case, after performing a measurement and getting a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In this situation, for each i, we know the interval $\mathbf{x}_i$ of possible values of x_i, and we need to find the range $\mathbf{y}$ of the function $f(x_1, \ldots, x_n)$ over all possible tuples $x_i \in \mathbf{x}_i$.

2.4 Interval Computations Are Sometimes Easy

In some cases, it is easy to estimate the desired range. For example, the arithmetic average E is a monotonically increasing function of each of its n variables $x_1, \ldots, x_n$, so its smallest possible value $\underline{E}$ is attained when each value x_i is the smallest possible ($x_i = \underline{x}_i$) and its largest possible value is attained when $x_i = \overline{x}_i$ for all i. In other words, the range $\mathbf{E}$ of E is equal to $[E(\underline{x}_1, \ldots, x_n), E(\overline{x}_1, \ldots, \overline{x}_n)]$, where,

$$\underline{E} = \frac{1}{n} \cdot (\underline{x}_1 + \ldots + \underline{x}_n)$$

and

$$\overline{E} = \frac{1}{n} \cdot (\overline{x}_1 + \ldots + \overline{x}_n).$$

2.5 Interval Computations Are, in General, Computationally Difficult

For more complex functions $C(x_1, \ldots, x_n)$, the problem of computing the range is often more computationally difficult.

For example, it is known that the problem of computing the exact range $\mathbf{V} = [\underline{V}, \overline{V}]$ for the variance V over interval data $x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ is, in general, NP-hard; see, e.g., [17, 18]. Specifically, there is a polynomial-time algorithm for computing $\underline{V}$, but computing $\overline{V}$ is, in general, NP-hard.

There are known algorithms for computing the range under interval uncertainty, algorithms that work in many practical situations; see, e.g., [14]. Let us briefly describe these methods and explain why they are not sufficient for statistical data processing under interval uncertainty (in particular, under privacy-related interval uncertainty).

2.6 Interval Arithmetic

2.6.1 Computing the Range of a Monotonic Function is Easy

For some functions $f(x_1, \ldots, x_n)$, computing the range

$$[\underline{y}, \overline{y}] = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{f(x_1, \ldots, x_n) : x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, x_n \in [\underline{x}_n, \overline{x}_n]\}$$

is reasonably easy. In particular, the computation of this range is easy if the function $f(x_1, \ldots, x_n)$ is monotonic with respect to each of its variables x_i.

Indeed, if a function $f(x_1, \ldots, x_i, \ldots, x_n)$ is an increasing function of the variable x_i, then:

- the function f attains its largest value on the interval $[\underline{x}_i, \overline{x}_i]$ is attained when x_i is the largest, i.e., when $x_i = \overline{x}_i$;
- the function f attains its smallest value on the interval $[\underline{x}_i, \overline{x}_i]$ is attained when x_i is the smallest, i.e., when $x_i = \underline{x}_i$.

Similarly, if a function $f(x_1, \ldots, x_i, \ldots, x_n)$ is a decreasing function of the variable x_i, then:

- the function f attains its largest value on the interval $[\underline{x}_i, \overline{x}_i]$ is attained when x_i is the smallest, i.e., when $x_i = \underline{x}_i$;
- the function f attains its smallest value on the interval $[\underline{x}_i, \overline{x}_i]$ is attained when x_i is the largest, i.e., when $x_i = \overline{x}_i$.

So, if a function $f(x_1, \ldots, x_n)$ is monotonic with respect to all its variable, then we can find $\overline{y}$ by applying f to values $x_i = \overline{x}_i$ for all i for which f is increasing and to values $x_i = \underline{x}_i$ for all i for which f is decreasing. Similarly, we can find $\underline{y}$ by applying f to values $x_i = \underline{x}_i$ for all i for which f is increasing and to values $x_i = \overline{x}_i$ for all i for which f is decreasing.

2.6.2 Interval Addition and Interval Subtraction

Arithmetic operations are examples of such monotonic functions. For example, addition $f(x_1, x_2) = x_1 + x_2$ is increasing in both variables, and subtraction $f(x_1, x_2) = x_1 - x_2$ is increasing as a function of the first variable x_1 and decreasing as a function of the second variable x_2. Thus, we can compute the range of these functions over intervals $[\underline{x}_1, \overline{x}_1]$ and $[\underline{x}_2, \overline{x}_2]$ as follows. For $f(x_1, x_2) = x_1 + x_2$, we have

$$[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2],$$

and for $f(x_1, x_2) = x_1 - x_2$, we have

$$[\underline{x}_1, \overline{x}_1] - [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2].$$

These operations are in full agreement with common sense. For example, if in one warehouse we have between 6.0 and 8.0 tons of some material, and in another warehouse, we have between 3.0 and 5.0 tons, then overall, in the worst case, we can have $6.0 + 3.0 = 9.0$ tons (if both are at their lowest levels), and in the best case, we can have up to $8.0 + 5.0 = 13.0$ tons (when both are at their largest levels).

Similarly, if a warehouse originally had between 6.0 and 8.0 tons, and we moved between 1.0 and 2.0 tons to another location, then the smallest amount left is when we start with the smallest possible value 6.0 and move the largest possible value 2.0, resulting in $6.0 - 2.0 = 4.0$. The largest amount left is when we start with the largest possible value 8.0 and move the smallest possible value 1.0, resulting in $8.0 - 1.0 = 7.0$.

2.6.3 Interval Multiplication

The situation with product $f(x_1, x_2) = x_1 \cdot x_2$ is slightly more complex. The product function is monotonic with respect to each of its variables, so we know that the largest possible value of this function on given intervals $[\underline{x}_1, \overline{x}_1]$ and $[\underline{x}_2, \overline{x}_2]$ is attained when each of the variables x_i attains one of its endpoints: $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$. However, in contrast to the cases of addition and substraction, it is not clear where exactly this maximum is attained. Indeed, with respect to x_1, the product function is increasing when $x_2 \geq 0$ and decreasing when $x_2 \leq 0$. Since the interval $[\underline{x}_2, \overline{x}_2]$ may contain both negative and positive values of x_2, we cannot beforehand know whether the largest possible value will be at $x_1 = \underline{x}_1$ or at $x_1 = \overline{x}_1$. One way to solve this problem is to consider both possible values of x_1 ($x_1 = \underline{x}_1$ and $x_1 = \overline{x}_1$), and for each of them, consider both possible values of x_2 ($x_2 = \underline{x}_2$ and $x_2 = \overline{x}_2$). Thus, we have $2 \times 2 = 4$ possible combinations and we know that the maximum and the minimum of the product function are attained at one of these combinations. So, we arrive at the following formula:

$$[\underline{x}_1, \overline{x}_1] \cdot [\underline{x}_2, \overline{x}_2] = [\min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2), \max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2)].$$

2.6.4 Interval Division

For division $f(x_1, x_2) = \dfrac{x_1}{x_2}$, the situation is similar: the corresponding function is monotonic in x_1 and in x_2, so, we can find an explicit expression for its range. The only additional difficulty here is that we must make sure that the fraction $\dfrac{x_1}{x_2}$ is defined for all values x_2 from the interval $[\underline{x}_2, \overline{x}_2]$, i.e., that this interval does not contain 0. The formula for the interval division can be simplified if we take into account that in most modern computers, division is implemented as a product $x_1 \cdot \dfrac{1}{x_2}$. The operation $\dfrac{1}{x_2}$ is decreasing in x_2, so its range can be easily found: if $0 \notin [\underline{x}_2, \overline{x}_2]$, then

$$\frac{1}{[\underline{x}_2, \overline{x}_2]} = \left[\frac{1}{\overline{x}_2}, \frac{1}{\underline{x}_2}\right]$$

and

$$\frac{[\underline{x}_1, \overline{x}_1]}{[\underline{x}_2, \overline{x}_2]} = [\underline{x}_1, \overline{x}_1] \cdot \frac{1}{[\underline{x}_2, \overline{x}_2]}.$$

2.6.5 Additional Operations

In the computers, several other operations are often hardware supported. For such operations, we can often also compute the exact range. For example, the function $f(x_1, x_2) = \max(x_1, x_2)$ in (non-strictly) increasing in both x_1 and x_2, so its range can be computed as

$$\max([\underline{x}_1, \overline{x}_1], [\underline{x}_2, \overline{x}_2]) = [\max(\underline{x}_1, \underline{x}_2), \max(\overline{x}_1, \overline{x}_2)].$$

Similarly, the function $f(x_1, x_2) = \min(x_1, x_2)$ in (non-strictly) increasing in both x_1 and x_2, so

$$\min([\underline{x}_1, \overline{x}_1], [\underline{x}_2, \overline{x}_2]) = [\min(\underline{x}_1, \underline{x}_2), \min(\overline{x}_1, \overline{x}_2)].$$

The function $f(x_1) = x_2^2$ is increasing when $x_1 \geq 0$ and decreasing when $x_1 \leq 0$, so its range can be computed as follows:

- if $0 \leq \underline{x}_1$, then

$$[\underline{x}_1, \overline{x}_1]^2 = [\underline{x}_1^2, \overline{x}_1^2];$$

- if $\overline{x}_1 \leq 0$, then

$$[\underline{x}_1, \overline{x}_1]^2 = [\overline{x}_1^2, \underline{x}_1^2];$$

- finally, if $\underline{x}_1 < 0 < \overline{x}_1$, then

$$[\underline{x}_1, \overline{x}_1]^2 = [0, \max(\underline{x}_1^2, \overline{x}_1^2)].$$

Similarly, for every even value $2p$, the function x_1^{2p} is increasing when when $x_1 \geq 0$ and decreasing when $x_1 \leq 0$, so

- if $0 \leq \underline{x}_1$, then

$$[\underline{x}_1, \overline{x}_1]^{2p} = [\underline{x}_1^{2p}, \overline{x}_1^{2p}];$$

- if $\overline{x}_1 \le 0$, then

$$[\underline{x}_1, \overline{x}_1]^{2p} = [\overline{x}_1^{2p}, \underline{x}_1^{2p}];$$

- finally, if $\underline{x}_1 < 0 < \overline{x}_1$, then

$$[\underline{x}_1, \overline{x}_1]^{2p} = [0, \max(\underline{x}_1^{2p}, \overline{x}_1^{2p})].$$

For odd values $2p+1$, the function x_1^{2p+1} is increasing in x_1, so

$$[\underline{x}_1, \overline{x}_1]^{2p+1} = [\underline{x}_1^{2p+1}, \overline{x}_1^{2p+1}].$$

2.7 Straightforward Interval Computations

2.7.1 Main Idea

We know how to compute the range for each arithmetic operation. Therefore, to compute the range $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, it is reasonable to do the following:

- first, we parse the algorithm f, i.e., represent it as a sequence of elementary arithmetic operations (this is done automatically by a compiler);
- then, we repeat the computations forming the program f step-by-step, replacing each operation with real numbers by the corresponding operation of interval arithmetic.

It is known that, as a result, we get an enclosure $\mathbf{Y}$ for the desired range $\mathbf{y}$.

2.7.2 Example When Straightforward Interval Computations Work Perfectly

Let us start with an example of computing the average of two values

$$f(x_1, x_2) = 0.5 \cdot (x_1 + x_2).$$

This function is increasing in both variables, so its range on the intervals $[\underline{x}_1, \overline{x}_1]$ and $[\overline{x}_2, \overline{x}_2]$ is equal to $[0.5 \cdot (\underline{x}_1 + \underline{x}_2), 0.5 \cdot (\overline{x}_1 + \overline{x}_2)]$.

A compiler will parse the function f into the following sequence of computational steps:

- we start with x_1 and x_2;
- then, we compute an intermediate value $x_3 = x_1 + x_2$;
- finally, we compute $y = 0.5 \cdot x_3$.

According to straightforward interval computations:

- we start with $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$ and $\mathbf{x}_2 = [\underline{x}_2, \overline{x}_2]$;
- then, we compute $\mathbf{x}_3 = \mathbf{x}_1 + \mathbf{x}_2 = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2]$;
- finally, we compute $\mathbf{y} = 0.5 \cdot \mathbf{x}_3$, and we get the desired range.

One can easily check that we also get the exact range for the general case of the arithmetic average (i.e., population average), and, even more generally, for an arbitrary linear function $f(x_1, \ldots, x_n)$.

2.7.3 Can Straightforward Interval Computations be Always Perfect?

In straightforward interval computations, we replace each elementary arithmetic operation with the corresponding operation of interval arithmetic. We have already mentioned that this replacement increases the computation time at most by a factor of 4. So, if we started with the polynomial time, we still get polynomial time.

On the other hand, we know that the main problem of interval computations is NP-hard. This means, crudely speaking, that we cannot always compute the exact range by using a polynomial-time algorithm. Since straightforward interval computations is a

polynomial-time algorithm, this means that in some cases, its estimates for the range are not exact.

Let us describe a simple example when this happens.

2.7.4 Case of Population Variance

For the problem of computing the range of finite population variance, the situation is somewhat more difficult. Let us consider a simple example of two interval values, with $\mathbf{x}_1 = \mathbf{x}_2 = [0, 1]$. For two values, the formula for the variance V can be simplified into $V = (x_1 - x_2)^2/4$. Since $x_1 \in [0, 1]$ and $x_2 \in [0, 1]$, the range of possible values of the difference $x_1 - x_2$ can be determined by interval arithmetic as $[0, 1] - [0, 1] = [-1, 1]$. Thus, the actual range of $(x_1 - x_2)^2$ is $[0, 1]$, and the actual range for the variance $V = (x_1 - x_2)^2/4$ is $[0, 1]/4 = [0, 0.25]$.

Let us show that for this problem, straightforward interval computations do not lead to the exact range. Indeed, the compiler will parse the expression $V = \dfrac{(x_1 - E)^2 + (x_2 - E)^2}{2}$, where $E = \dfrac{x_1 + x_2}{2}$, into the following sequence of elementary operations:

- we start with x_1 and x_2;
- we compute an intermediate value $x_3 = x_1 + x_2$;
- we compute an intermediate value $E = 0.5 \cdot x_3$;
- we compute an intermediate value $x_4 = x_1 - E$;
- we compute an intermediate value $x_5 = x_2 - E$;
- we compute an intermediate value $x_6 = x_4^2$;
- we compute an intermediate value $x_7 = x_5^2$;
- we compute an intermediate value $x_8 = x_6 + x_7$;
- finally, we compute $V = 0.5 \cdot x_8$.

According to the main idea of straightforward interval computations, we replace each elementary operation with the corresponding operation from interval arithmetic:

- we start with $\mathbf{x}_1 = [0, 1]$ and $\mathbf{x}_2 = [0, 1]$;
- we compute $\mathbf{x}_3 = \mathbf{x}_1 + \mathbf{x}_2 = [0, 1] + [0, 1] = [0 + 0, 1 + 1] = [0, 2]$;
- we compute $\mathbf{E} = 0.5 \cdot \mathbf{x}_3 = 0.5 \cdot [0, 2] = [0.5 \cdot 0, 0.5 \cdot 2] = [0, 1]$;
- we compute $\mathbf{x}_4 = \mathbf{x}_1 - \mathbf{E} = [0, 1] - [0, 1] = [0 - 1, 1 - 0] = [-1, 1]$;
- we compute $\mathbf{x}_5 = \mathbf{x}_2 - \mathbf{E} = [0, 1] - [0, 1] = [0 - 1, 1 - 0] = [-1, 1]$;
- we compute $\mathbf{x}_6 = \mathbf{x}_4^2 = [-1, 1]^2 = [0, \max((-1)^2, 1^2)] = [0, 1]$;
- we compute $\mathbf{x}_7 = \mathbf{x}_5^2 = [-1, 1]^2 = [0, \max((-1)^2, 1^2)] = [0, 1]$;
- we compute $\mathbf{x}_8 = \mathbf{x}_6 + \mathbf{x}_7 = [0, 1] + [0, 1] = [0 + 0, 1 + 1] = [0, 2]$;
- finally, we compute $\mathbf{V} = 0.5 \cdot \mathbf{x}_8 = 0.5 \cdot [0, 2] = [0.5 \cdot 0, 0.5 \cdot 2] = [0, 1]$.

The resulting interval $[0, 1]$ is an enclosure for the actual range $[0, 0.25]$: $[0, 1] \supset [0, 0.25]$, but it is wider than the actual range, i.e., it has *excess width*.

It is worth mentioning that there are other formulas one can use to compute the variance of a finite population: e.g., the formula

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - E^2.$$

However, this formula also leads to excess width for the above example. Indeed, in this example, $V = \frac{x_1^2 + x_2^2}{2} - E^2$, so the compiler will parse this formula into the following sequence of elementary operations:

- we start with x_1 and x_2;
- we compute an intermediate value $x_3 = x_1 + x_2$;

- we compute an intermediate value $E = 0.5 \cdot x_3$;
- we compute an intermediate value $x_4 = x_1^2$;
- we compute an intermediate value $x_5 = x_2^2$;
- we compute an intermediate value $x_6 = x_4 + x_5$;
- we compute an intermediate value $x_7 = 0.5 \cdot x_6$;
- we compute an intermediate value $x_8 = E^2$;
- finally, we compute $V = x_7 - x_8$.

So, the straightforward interval computations result in the following sequence of elementary operations:

- we start with $\mathbf{x}_1 = [0, 1]$ and $\mathbf{x}_2 = [0, 1]$;
- we compute $\mathbf{x}_3 = \mathbf{x}_1 + \mathbf{x}_2 = [0, 1] + [0, 1] = [0 + 0, 1 + 1] = [0, 2]$;
- we compute $\mathbf{E} = 0.5 \cdot \mathbf{x}_3 = 0.5 \cdot [0, 2] = [0.5 \cdot 0, 0.5 \cdot 2] = [0, 1]$;
- we compute $\mathbf{x}_4 = \mathbf{x}_1^2 = [0, 1]^2 = [0^2, 1^2] = [0, 1]$;
- we compute $\mathbf{x}_5 = \mathbf{x}_2^2 = [0, 1]^2 = [0^2, 1^2] = [0, 1]$;
- we compute $\mathbf{x}_6 = \mathbf{x}_4 + \mathbf{x}_5 = [0, 1] + [0, 1] = [0 + 0, 1 + 1] = [0, 2]$;
- we compute $\mathbf{x}_7 = 0.5 \cdot \mathbf{x}_6 = 0.5 \cdot [0, 2] = [0.5 \cdot 0, 0.5 \cdot 2] = [0, 1]$;
- we compute $\mathbf{x}_8 = \mathbf{E}^2 = [0, 1]^2 = [0^2, 1^2] = [0, 1]$;
- finally, we compute $\mathbf{V} = \mathbf{x}_7 - \mathbf{x}_8 = [0, 1] - [0, 1] = [0 - 1, 1 - 0] = [-1, 1]$.

Here, we get an even wider interval, with more excess width.

In this formula too, each variable x_i occurs several times, as a result of which we get excess width: for $\mathbf{x}_1 = \mathbf{x}_2 = [0, 1]$, we get $\boldsymbol{\mu} = [0, 1]$ and

$$\frac{\mathbf{x}_1^2 + \mathbf{x}_2^2}{2} - \boldsymbol{\mu}^2 = [-1, 1] \supset [0, 0.25].$$

2.7.5 Interval Computations go Beyond Straightforward Technique

It is sometimes erroneously assumed that the above straightforward ("naive") techniques is all there is in interval computations. In reality, interval computations is *not a single algorithm*, it is a *problem* for which many different techniques exist. Let us now describe some of such techniques.

2.8 Centered Form

2.8.1 Main Idea

One of the advanced interval computations techniques is the centered form technique. This technique is based on the Taylor series expansion of the corresponding function $f(x_1, \ldots, x_n)$.

We start by representing each interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ in the form $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$, where $\widetilde{x}_i = (\underline{x}_i + \overline{x}_i)/2$ is the midpoint of the interval $\mathbf{x}_i$ and $\Delta_i = (\overline{x}_i - \underline{x}_i)/2$ is the half-width of this interval.

After that, we use the Taylor expansion. Specifically, the centered form is based on the formula

$$f(x_1, \ldots, x_n) = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\eta_1, \ldots, \eta_n) \cdot (x_i - \widetilde{x}_i),$$

where each η_i is some value from the interval $\mathbf{x}_i$.

Since $\eta_i \in \mathbf{x}_i$, the value of the i-th derivative belongs to the interval range of this derivative on these intervals. We also know that $x_i - \widetilde{x}_i \in [-\Delta_i, \Delta_i]$. Thus, we can conclude that

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subseteq f(\widetilde{x}_1, \ldots, \widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \cdot [-\Delta_i, \Delta_i].$$

To compute the ranges of the partial derivatives, we can use straightforward interval computations.

2.8.2 Example

Let us illustrate the centered form techniques on the above example of computing the population variance.

When all the intervals are the same, e.g., when $\mathbf{x}_i = [0, 1]$, the centered form does not lead to the exact range. Indeed, the centered form always produces an interval centered at the point $f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$. In this case, all midpoints $\widetilde{x}_i$ are the same (e.g., equal to 0.5), hence the finite population variance $f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is equal to 0 on these midpoints. Thus, as a result of applying the centered form, we get an interval centered at 0, i.e., the interval whose lower endpoint is negative. In reality, the variance V is always non-negative, so negative values of V are impossible.

The upper endpoint produced by the centered form is also different from the upper endpoint of the actual range: e.g., for $\mathbf{x}_1 = \mathbf{x}_2 = [0, 1]$, we have $\frac{\partial f}{\partial x_1}(x_1, x_2) = (x_1 - x_2)/2$, hence

$$\frac{\partial f}{\partial x_1}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 - \mathbf{x}_2}{2} = [-0.5, 0.5].$$

A similar formula holds for the derivative with respect to x_2. Since $\Delta_i = 0, 5$, the centered form leads to:

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subseteq 0 + [-0.5, 0.5] \cdot [-0.5, 0.5] + [-0.5, 0.5] \cdot [-0.5, 0.5] = [-0.5, 0.5]$$

– an excess width in comparison with the actual range $[0, 0.25]$.

2.8.3 How Can We Get Better Estimates?

In the centered form, we, in effect, ignored quadratic and higher order terms, i.e., terms of the type

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \cdot \Delta x_i \cdot \Delta x_j.$$

When the estimate is not accurate enough, it means that either this ignored term (or some other ignored term) is too large. There are two ways to reduce the size of the ignored term:

- we can try to decrease the value of this term, or

- we can try to explicitly include higher order terms in the Taylor expansion formula, so that the remainder term will be proportional to, say, Δx_i^3 and thus, be much smaller.

Let us describe these two approaches in detail.

2.9 First Approach: Bisection

2.9.1 Main Idea

Let us first describe the situation in which we try to minimize the second-order remainder term. In the above expression for this term, we cannot change the second derivative, it remains, crudely speaking, of the same order of magnitude, bounded by the same bound. The only thing we can drastically decrease is the difference $\Delta x_i = x_i - \widetilde{x}_i$ between the actual value and the midpoint. This value is bounded by the half-width Δ_i of the box. So, to decrease this value, we can subdivide the original box into several narrower subboxes. Usually, we divide into two subboxes, so this subdivision is called *bisection*.

The range over the whole box is equal to the union of the ranges over all the subboxes. The widths of each subbox are smaller, so we get smaller Δx_i and hopefully, more accurate estimates for ranges over each of this subbox. Then, we take the union of the ranges over subboxes.

2.9.2 Example

Let us bisect the first interval $\mathbf{x}_1 = [0, 1]$ into two half-intervals $\mathbf{x}_1 = [0, 0.5]$ and $\mathbf{x}_1 = [0.5, 1]$. Then, according to the bisection method, we compute the range as follows:

- first, we use the centered form to compute an enclosure for V on $\mathbf{x}_1 = [0, 0.5]$ and $\mathbf{x}_2 = [0, 1]$;
- second, we use the centered form to compute an enclosure for V on $\mathbf{x}_1 = [0.5, 1]$ and $\mathbf{x}_2 = [0, 1]$;

- finally, we take the union of the two enclosures.

Let us first describe how we can use the centered form for $\mathbf{x}_1 = [0, 0.5]$ and $\mathbf{x}_2 = [0, 1]$. In this case, $\widetilde{x}_1 = (0 + 0.5)/2 = 0.25$ and $\Delta_1 = (0.5 - 0)/2 = 0.25$, so

$$\widetilde{E} = \frac{\widetilde{x}_1 + \widetilde{x}_2}{2} = \frac{0.25 + 0.5}{2} = 0.375,$$

and

$$f(\widetilde{x}_1, \widetilde{x}_2) = \frac{(\widetilde{x}_1 - \widetilde{E})^2 + (\widetilde{x}_2 - \widetilde{E})^2}{2} = \frac{0.125^2 + 0.125^2}{2} = 0.125^2 = 0.015625.$$

Here,

$$\frac{\partial f}{\partial x_1}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 - \mathbf{x}_2}{2} = \frac{[0, 0.5] - [0, 1]}{2} = \frac{[-1, 0.5]}{2} = [-0.5, 0.25],$$

and

$$\frac{\partial f}{\partial x_2}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_2 - \mathbf{x}_1}{2} = \frac{[0, 1] - [0, 0.5]}{2} = \frac{[-0.5, 1]}{2} = [-0.25, 0.5],$$

so

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subseteq 0.015625 + [-0.5, 0.25] \cdot [-0.25, 0.25] + [-0.25, 0.5] \cdot [-0.5, 0.5] =$$

$$0.015625 + [-0.125, 0.125] + [-0.25, 0.25] = [-0.359375, 0.390625].$$

Similarly, for $\mathbf{x}_1 = [0.5, 1]$ and $\mathbf{x}_2] = [0, 1]$, we have $\widetilde{x}_1 = (0.5 + 1)/2 = 0.75$ and $\Delta_1 = (1 - 0.5)/2 = 0.25$, so

$$\widetilde{E} = \frac{\widetilde{x}_1 + \widetilde{x}_2}{2} = 0.75 + 0.52 = 0.625,$$

and

$$f(\widetilde{x}_1, \widetilde{x}_2) = \frac{(\widetilde{x}_1 - \widetilde{E})^2 + (\widetilde{x}_2 - \widetilde{E})^2}{2} = \frac{0.125^2 + 0.125^2}{2} = 0.125^2 = 0.015625.$$

Here,

$$\frac{\partial f}{\partial x_1}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 - \mathbf{x}_2}{2} = \frac{[0.5, 1] - [0, 1]}{2} = \frac{[-0.5, 1]}{2} = [-0.25, 0.5],$$

and

$$\frac{\partial f}{\partial x_2}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_2 - \mathbf{x}_1}{2} = \frac{[0, 1] - [0.5, 1]}{2} = \frac{[-1, 0.5]}{2} = [-0.5, 0.25],$$

so

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subseteq 0.015625 + [-0.25, 0.5] \cdot [-0.25, 0.25] + [-0.5, 0.25] \cdot [-0.5, 0.5] =$$

$$0.015625 + [-0.125, 0.125] + [-0.25, 0.25] = [-0.359375, 0.390625].$$

In this example, the two enclosures coincide, so their union is equal to the same interval $[-0.359375, 0.390625]$. The resulting enclosure is narrower than the original one $[-0.5, 0.5]$ but still has excess width in comparison with the actual range $[0, 0.25]$.

2.9.3 Bisection: General Comment

The more subboxes we consider, the smaller Δx_i and thus, the more accurate the corresponding enclosures. However, once we have more boxes, we need to spend more time processing these boxes. Thus, we have a trade-off between computation time and accuracy: the more computation time we allow, the more accurate estimates we will be able to compute.

2.9.4 Additional Idea: Monotonicity Checking

If the function $f(x_1, \ldots, x_n)$ is monotonic over the original box $\mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, then we can easily compute its exact range. Since we used the centered form for the original box, this probably means that on that box, the function is not monotonic: for example, with respect to x_1, it may be increasing at some points in this box, and decreasing at other points.

However, as we divide the original box into smaller subboxes, it is quite possible that at least some of these subboxes will be outside the areas where the derivatives are 0 and thus, the function $f(x_1, \ldots, x_n)$ will be monotonic. So, after we subdivide the box into subboxes, we should first check monotonicity on each of these subboxes – and if the function is monotonic, we can easily compute its range.

In calculus terms, a function is increasing with respect to x_i if its partial derivative $\dfrac{\partial f}{\partial x_i}$ is non-negative everywhere on this subbox. Thus, to check monotonicity, we should find

the range $[\underline{y}_i, \overline{y}_i]$ of this derivative (we need to do it anyway to compute the centered form expression):

- if $\underline{y}_i \geq 0$, this means that the derivative is everywhere non-negative and thus, the function f is increasing in x_i;
- if $\overline{y}_i \leq 0$, this means that the derivative is everywhere non-positive and thus, the function f is decreasing in x_i.

If $\underline{y}_i < 0 < \overline{y}_i$, then we have to use the centered form.

If the function is monotonic (e.g., increasing) only with respect to some of the variables x_i, then

- to compute $\overline{y}$, it is sufficient to consider only the value $x_i = \overline{x}_i$, and
- to compute $\underline{y}$, it is sufficient to consider only the value $x_i = \underline{x}_i$.

For such subboxes, we reduce the original problem to two problems with fewer variables, problems which are thus easier to solve.

2.9.5 Example

In the above example, monotonicity works when we bisect each of the two input intervals $\mathbf{x}_1 = \mathbf{x}_2 = [0, 1]$ into two halves $[0, 0.5]$ and $[0, 5, 1]$. In this case, we get four boxes $\mathbf{x}_1 \times \mathbf{x}_2$:

- $\mathbf{x}_1 = [0, 0.5]$ and $\mathbf{x}_2 = [0, 0.5]$;
- $\mathbf{x}_1 = [0, 0.5]$ and $\mathbf{x}_2 = [0.5, 1]$;
- $\mathbf{x}_1 = [0.5, 1]$ and $\mathbf{x}_2 = [0, 0.5]$;
- $\mathbf{x}_1 = [0.5, 1]$ and $\mathbf{x}_2 = [0.5, 1]$.

In the second case $\mathbf{x}_1 = [0, 0.5]$ and $\mathbf{x}_2 = [0.5, 1]$, the range of the first derivative is equal to

$$\frac{\partial f}{\partial x_1}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 - \mathbf{x}_2}{2} = \frac{[0, 0.5] - [0.5, 1]}{2} = \frac{[-1, 0]}{2} = [-0.5, 0],$$

so we can guarantee that on this box, the variance is a decreasing function of x_1. Similarly,

$$\frac{\partial f}{\partial x_2}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_2 - \mathbf{x}_1}{2} = \frac{[0.5, 1] - [0, 0.5]}{2} = \frac{[0, 1]}{2} = [0, 0.5],$$

so we can guarantee that on this box, the variance is an increasing function of x_2. Thus, we can compute the exact range of $V = f(x_1, x_2)$ on this box as

$$[f(\overline{x}_1, \underline{x}_2), f(\underline{x}_1, \overline{x}_2)] = [f(0.5, 0.5), f(0, 1)].$$

For $x_1 = x_2 = 0.5$, we have $E = \dfrac{x_1 + x_2}{2} = \dfrac{0.5 + 0.5}{2} = 0.5$, and hence

$$V = \frac{(x_1 - E)^2 + (x_2 - E)^2}{2} = 0.$$

For $x_1 = 0$ and $x_2 = 1$, we have $E = \dfrac{x_1 + x_2}{2} = \dfrac{0 + 1}{2} = 0.5$, and hence

$$V = \frac{(x_1 - E)^2 + (x_2 - E)^2}{2} = \frac{(0 - 0.5)^2 + (1 - 0.5)^2}{2} = \frac{0.25 + 0.25}{2} = 0.25,$$

so this range is equal to $[0, 0.25]$.

The fourth case $\mathbf{x}_1 = [0.5, 1]$ and $\mathbf{x}_2 = [0, 0.5]$ is similar to the second one, with x_2 instead of x_1. Since the variance is a symmetric function, i.e., its value does not change when we swap x_1 and $x - 2$, its range also does not change if we swap intervals for x_1 and x_2. Thus, in the fourth case, we also get the range $[0, 0.25]$.

In the first case $\mathbf{x}_1 = \mathbf{x}_2 = [0, 0.5]$, we have $\widetilde{x}_1 = \widetilde{x}_2 = (0 + 0.5)/2 = 0.25$ and $\Delta_1 = \Delta x_2 = (0.5 - 0)/2 = 0.25$, so

$$\widetilde{E} = \frac{\widetilde{x}_1 + \widetilde{x}_2}{2} = 0.25 + 0.252 = 0.25,$$

and

$$f(\widetilde{x}_1, \widetilde{x}_2) = \frac{(\widetilde{x}_1 - \widetilde{E})^2 + (\widetilde{x}_2 - \widetilde{E})^2}{2} = 0.$$

Here,

$$\frac{\partial f}{\partial x_1}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 - \mathbf{x}_2}{2} = \frac{[0, 0.5] - [0, 0.5]}{2} = \frac{[-0.5, 0.5]}{2} = [-0.25, 0.25],$$

and

$$\frac{\partial f}{\partial x_2}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_2 - \mathbf{x}_1}{2} = \frac{[0, 0.5] - [0, 0.5]}{2} = \frac{[-0.5, 0.5]}{2} = [-0.25, 0.25],$$

so

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subseteq 0 + [-0.25, 0.25] \cdot [-0.25, 0.25] + [-0.25, 0.25] \cdot [-0.25, 0.25] =$$

$$0 + [-0.0625, 0.0625] + [-0.0625, 0.0625] = [-0.125, 0.125].$$

Similarly, in the third case $\mathbf{x}_1 = \mathbf{x}_2 = [0.5, 1]$, we have $\widetilde{x}_1 = \widetilde{x}_2 = (0.5 + 1)/2 = 0.75$ and $\Delta_1 = \Delta x_2 = (1 - 0.5)/2 = 0.25$, so

$$\widetilde{E} = \frac{\widetilde{x}_1 + \widetilde{x}_2}{2} = \frac{0.75 + 0.75}{2} = 0.75,$$

and

$$f(\widetilde{x}_1, \widetilde{x}_2) = \frac{(\widetilde{x}_1 - \widetilde{E})^2 + (\widetilde{x}_2 - \widetilde{E})^2}{2} = 0.$$

Here,

$$\frac{\partial f}{\partial x_1}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1 - \mathbf{x}_2}{2} = \frac{[0.5, 1] - [0.5, 1]}{2} = \frac{[-0.5, 0.5]}{2} = [-0.25, 0.25],$$

and

$$\frac{\partial f}{\partial x_2}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_2 - \mathbf{x}_1}{2} = \frac{[0.5, 1] - [0.5, 1]}{2} = \frac{[-0.5, 0.5]}{2} = [-0.25, 0.25],$$

so

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subseteq 0 + [-0.25, 0.25] \cdot [-0.25, 0.25] + [-0.25, 0.25] \cdot [-0.25, 0.25] =$$

$$[-0.125, 0.125].$$

The union of the resulting four enclosures

$$[0, 0.25], [0, 0.25], [-0.125, 0.125], [-0.125, 0.125]$$

is equal to $[-0.125, 0.25]$. The resulting enclosure still has excess width, but it is much closer to the actual range $[0, 0.25]$ – e.g., the upper bound is now exact.

2.10 General Taylor Techniques

2.10.1 Main Idea

As we have mentioned, another way to get more accurate estimates is to use so-called *Taylor techniques*, i.e., to explicitly consider second-order and higher-order terms in the Taylor expansion.

Let us illustrate the main ideas of Taylor analysis on the case when we allow second order terms. In this case, the formula with a remainder takes the form

$$f(x_1,\ldots,x_n) = f(\widetilde{x}_1,\ldots,\widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\widetilde{x}_1,\ldots,\widetilde{x}_n)\cdot(x_i-\widetilde{x}_i) +$$

$$\frac{1}{2}\cdot\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial^2 f}{\partial x_i \partial x_j}(\eta_1,\ldots,\eta_n)\cdot(x_i-\widetilde{x}_i)\cdot(x_j-\widetilde{x}_j).$$

Thus, we get the enclosure

$$f(\mathbf{x}_1,\ldots,\mathbf{x}_n) \subseteq f(\widetilde{x}_1,\ldots,\widetilde{x}_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\widetilde{x}_1,\ldots,\widetilde{x}_n)\cdot[-\Delta_i,\Delta_i] +$$

$$\frac{1}{2}\cdot\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}_1,\ldots,\mathbf{x}_n)\cdot[-\Delta_i,\Delta_i]\cdot[-\Delta_j,\Delta_j],$$

where for $i=j$, we have to use the formula for the interval square $[-\Delta_i,\Delta_i]^2 = [0,\Delta_i^2]$ instead of the more general formula for the interval multiplication.

2.10.2 Example

Let us illustrate this idea on the above example of the variance $f(x_1,x_2)=V$ for $\mathbf{x}_1 = \mathbf{x}_2 = [0,1]$. Here, $\widetilde{x}_1 = \widetilde{x}_2 = 0.5$ and $\Delta_1 = \Delta_2 = 0.5$, so $f(\widetilde{x}_1,\widetilde{x}_2) = 0$,

$$\frac{\partial f}{\partial x_1}(\widetilde{x}_1,\widetilde{x}_2) = \frac{\widetilde{x}_1-\widetilde{x}_2}{2} = 0,$$

and similarly,

$$\frac{\partial f}{\partial x_2}(\widetilde{x}_1,\widetilde{x}_2) = \frac{\widetilde{x}_2-\widetilde{x}_1}{2} = 0.$$

Here,

$$\frac{\partial^2 V}{\partial x_1^2} = \frac{\partial^2 V}{\partial x_2^2} = \frac{1}{4}, \quad \frac{\partial^2 V}{\partial x_1 \partial x_2} = \frac{\partial^2 V}{\partial x_1 \partial x_2} = -\frac{1}{4},$$

so the above enclosure takes the form

$$f(\mathbf{x}_1, \mathbf{x}_2) \subseteq \frac{1}{4} \cdot [-\Delta_1, \Delta_1]^2 + \frac{1}{4} \cdot [-\Delta_2, \Delta_2]^2 - \frac{1}{2} \cdot [-\Delta_1, \Delta_1] \cdot [-\Delta_2, \Delta_2] =$$

$$\frac{1}{4} \cdot [0, 0.25] + \frac{1}{4} \cdot [0, 0.25] - \frac{1}{2} \cdot [-0.25, 0.25] = [0.0.125] + [-0.125, 0.125] = [-0.125, 0.25].$$

2.10.3 Taylor Methods: General Comment

The more terms we consider in the Taylor expansion, the smaller the remainder term and thus, the more accurate the corresponding enclosures. However, once we have more terms, we need to spend more time computing these terms. Thus, for Taylor methods, we also have a trade-off between computation time and accuracy: the more computation time we allow, the more accurate estimates we will be able to compute.

Chapter 3

Applications of Interval Computations to Statistical Analysis: What Is Known

3.1 Privacy-Related Interval Computations: A Brief Reminder

As we have mentioned earlier, one way to maintain privacy in statistical databases is to replace the *exact* values x_i stored in these databases with the *intervals* $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ between the appropriate threshold. For example, if the actual age is 23, and the thresholds are 0, 10, 20, 30, ..., then instead of the exact age of 23 we only keep the interval $[20, 30]$ which contains the actual age. One of the main objectives of statistical databases is to perform statistical data processing, i.e., to compute the values of different statistical characteristics $C(x_1, \ldots, x_n)$. Different values $x_i \in \mathbf{x}_i$ lead, in general, to different values of the desired characteristic. We must therefore be able to compute the range

$$\mathbf{C} = C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{C(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

of possible value of the desired characteristic $C(x_1, \ldots, x_n)$.

This problem is a particular case of the general problem of *interval computations:* given an algorithmically defined function $f(x_1, \ldots, x_n)$ and n intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, find the range

$$f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{f(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

of the given function on the given intervals.

For simple functions like the population mean

$$E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i,$$

there are efficient algorithms for computing the corresponding interval range. However, it is known that in general, the interval computation problem is NP-hard. Moreover, it is NP-hard already for the case when the function $f(x_1, \ldots, x_n)$ describe such a simple statistical characteristic as population variance

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2.$$

While in general, the interval computations problem is NP-hard, there exist practically important situations when it is possible to efficiently compute the desired range $\mathbf{C}$. Let us describe these cases in detail.

3.2 Cases When Efficient Algorithms Are Possible for Statistical Analysis of Interval Data

Let us enumerate the main practically important cases in which there exist efficient algorithms for computing different statistical characteristics. Most of these cases are described in detail in [18].

It is worth mentioning that in [18], these cases are mainly described and illustrated for the case when the inputs come from measurements and the interval uncertainty reflects the inaccuracy of the corresponding measurements. However, from the purely mathematical and algorithmic viewpoint, these algorithms simply do not use any specific features of the measurement process, they simply assume that we know the values with interval uncertainty. Thus, we can also apply these algorithms to the situations of interest to us, when the interval uncertainty comes not from the measurement inaccuracy, but from the desire to preserve privacy.

3.2.1 First Class: Narrow Intervals

As we mentioned in the previous chapter, we can use Taylor series-based techniques to estimate the desired range. When the intervals are narrow, we can safely ignore quadratic and higher order terms in the corresponding Taylor expansion and use the linearized formulas instead. Since for narrow intervals, the problem is easy to compute, the computational complexity of interval computations comes from the fact that intervals are reasonably wide. In other words, when intervals are narrower, the problems are easier.

How can we describe the notion of "narrow intervals" in precise terms? One way to do it is as follows: the actual values $x_1, \ldots, x_n$ of the measured quantity are real numbers, so they are usually different. The data intervals $\mathbf{x}_i$ contain these values. When the intervals $\mathbf{x}_i$ surrounding the corresponding points x_i are narrow, these intervals do not have non-degenerate intersections (they may have a single common point): for every two intervals $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ and $\mathbf{x}_j = [\underline{x}_j, \overline{x}_j]$, we have $(\underline{x}_i, \overline{x}_i) \cap (\underline{x}_j, \overline{x}_j) = \emptyset$. When their widths becomes larger than the distance between the original values, the intervals start intersecting.

Thus, the ideal case of "narrow intervals" can be described as the case when no two intervals $\mathbf{x}_i$ have a non-degenerate intersection.

3.2.2 Second Class: Slightly Wider Intervals

Slightly wider intervals correspond to the situation when few intervals intersect, i.e., when for some integer K, no set of K intervals has a common intersection.

3.2.3 Third Class: Single Measuring Instrument

Since we want to find the exact range $\mathbf{C}$ of a statistic C, it is important not only that intervals are relatively narrow, it is also important that they are approximately of the same size: otherwise, if, say, Δx_i^2 is of the same order as Δx_j, we cannot meaningfully ignore Δx_i^2 and retain Δx_j. In other words, the interval data set should not combine high-accurate measurement results (with narrow intervals) and low-accurate results (with wide intervals):

all measurements should have been done by a single measuring instrument (or at least by several measuring instruments of the same type).

How can we describe this mathematically? A clear indication that we have two measuring instruments (MI) of different quality is that one interval is a proper subset of the other one: $[\underline{x}_i, \overline{x}_i] \subseteq (\underline{x}_j, \overline{x}_j)$.

So, if all pairs of non-degenerate intervals satisfy the following *subset property* $[\underline{x}_i, \overline{x}_i] \not\subseteq (\underline{x}_j, \overline{x}_j)$, we say that the measurements were done *by a single MI.*

Comment. This restriction only refers to inexact measurement results, i.e., to non-degenerate intervals. In addition to such interval values, we may have exact values (degenerate intervals). For example, in geodetic measurements, we may select some point ("benchmark") as a reference point, and describe, e.g., elevation of each point relative to this benchmark. For the benchmark point itself, the relative elevation will be therefore exactly equal to 0. When we want to compute the variance of elevations, we want to include the benchmark point too. From this viewpoint, when we talk about measurements made by a single measuring instrument, we may allow degenerate intervals (i.e., exact numbers) as well.

A reader should be warned that in several published algorithms describing a single MI case [38], only non-degenerate intervals are considered. However, as one can easily see from these published proofs (and from the idea of these proofs, as described below), these algorithms can be easily modified to incorporate possible exact values x_i.

3.2.4 Fourth Class: Same Accuracy Measurement

In some situations, it is also reasonable to consider a specific case of the single MI case when all measurements are performed with exactly the same accuracy, i.e., in mathematical terms, when all non-degenerate intervals $[\underline{x}_i, \overline{x}_i]$ have exactly the same half-width $\Delta_i = \frac{1}{2} \cdot (\overline{x}_i - \underline{x}_i)$.

3.2.5 Fifth Class: Several MI

After the single MI case, the natural next case is when we have several MI, i.e., when our intervals are divided into several subgroups each of which has the above-described subset property.

3.2.6 Sixth Class: Privacy Case

Although these definitions are in terms of measurements, they make sense for other sources of interval data as well. For example, for privacy data, intervals either coincide (if the value corresponding to the two patients belongs to the same range) or are different, in which case they can only intersect in one point. Similarly to the above situation, we also allow exact values in addition to ranges; these values correspond, e.g., to the exact records made in the past, records that are already in the public domain.

We will call interval data with this property – that every two non-degenerate intervals either coincide or do nor intersect – *privacy case.*

Comment. For the privacy case, the subset property is satisfied, so algorithms that work for a single MI case work for the privacy case as well.

3.2.7 Seventh Class: Non-Detects

Similarly, if the only source of interval uncertainty is detection limits, i.e., if every measurement result is either an exact value or a *non-detect*, i.e., an interval $[0, DL_i]$ for some real number DL_i (with possibly different detection limits for different sensors), then the resulting non-degenerate intervals also satisfy the subset property. Thus, algorithms that work for a single MI case work for this "non-detects" case as well.

When all sensors have the same detection limit DL, this case becomes a particular case of the privacy one. Thus, algorithms that works for the general privacy case also works for this subcase of the non-detects case.

3.2.8 Summary

These classes can be summarized in the following table. In this table, the first row corresponds to a general case, other rows correspond to different classes of problems described in this section:

class number	class description
0	general case
1	narrow intervals: no intersection
2	slightly wider intervals $\leq K$ intervals intersect
3	single measuring instrument (MI): subset property – no interval is a "proper" subset of the other
4	same accuracy measurements: all intervals have the same half-width
5	several (m) measuring instruments: intervals form m groups, with subset property in each group
6	privacy case: intervals same or non-intersecting
7	non-detects case: only non-degenerate intervals are $[0, DL_i]$

Table 3.1: Interval data: the general class and practically important subclasses

3.3 Computational Complexity of Statistical Analysis of Interval Data: What Is Known

Following [18], let us describe what is known about the computational complexity of statistical analysis of interval data. The following table describes, for different classes of interval data from Table 3.1, the computational complexity of computing the range of the standard statistical characteristics: the population mean

$$E = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i,$$

the population variance

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2,$$

the population covariance

$$C_{xy} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \cdot y_i,$$

the bounds $L = E - k_0 \cdot \sigma$ and $R = E + k_0 \cdot \sigma$ of the "k_0-sigma" interval, where $\sigma \stackrel{\text{def}}{=} \sqrt{V}$ and $k_0 = 2$, 3, or 6, and central moments

$$M_m = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^m$$

corresponding to even $m = 2p$ and odd $m = 2p + 1$ values m.

#	E	V	C_{xy}	L, U	M_{2p}	M_{2p+1}
0	$\Theta(n)$	NP-hard	NP-hard	NP-hard	NP-hard	?
1	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
2	$\Theta(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
3	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?
4	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?
5	$\Theta(n)$	$\mathcal{O}(n^{m+1})$	?	$\mathcal{O}(n^{m+1})$	$\mathcal{O}(n^{m+1})$	?
6	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?
7	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?

Table 3.2: Computational complexity of statistical analysis of interval data

As we can see, the main missing case is the case of the odd central moments. This is the case that we will handle in this thesis.

Before we describe our new algorithm for this case, let us describe, in detail, the known algorithms for computing second and third central moments under privacy-related interval uncertainty.

Comment. One might be surprised the fact that, for example, in the first row, computing E requires linear time, computing V (and hence $\sigma = \sqrt{V}$) requires time $\mathcal{O}(n \cdot \log(n))$, but computing the linear combination $U = E + k_0 \cdot \sigma$ requires quadratic time. This would have been strange if we were talking about computing the *values* of the corresponding characteristics: indeed, if we can compute E in time $\mathcal{O}(n)$ and σ in time $\mathcal{O}(n \cdot \log(n))$, then we can use these results to compute their linear combination U in time $\mathcal{O}(n \cdot \log(n))$. However, we are not computing the *values*, we are computing the *ranges* of the characteristics, and it is well known that a range of the sum $f(x) + g(x)$ of two functions cannot, in general, be computed if we only know the ranges for $f(x)$ and $g(x)$.

For example, if $f(x) = g(x) = x$, then on the interval $[0, 1]$, the range of both functions

is $[0, 1]$, and the range of the sum $f(x) + g(x) = 2x$ is $[0, 2]$. On the other hand, if $f(x) = x$ and $g(x) = 1 - x$, then the ranges of both functions $f(x)$ and $g(x)$ on the interval $[0, 1]$ are still equal to $[0, 1]$, but here $f(x) + g(x) = 1$ and so the range of the sum $f(x) + g(x)$ consists of a single point 1.

The answer is that we are *not* computing the values

It is worth mentioning that the range of the sum of the two function is not equal to the sum of their ranges.

Chapter 4

Statistical Analysis of Privacy-Related Interval Data: Known Algorithms

In the previous chapter, we mentioned several class of interval data for which efficient statistical analysis algorithms are possible. In this chapter, we present such algorithms for the single MI class (subset property) class – which includes privacy-related class as a particular case. We need to present these algorithms because the new algorithms that we develop in this thesis are based on these known ones.

While the algorithms and their justifications are known, we have somewhat modified their descriptions and made them more detailed. The reason for these modification is that our objective is to develop new algorithms based on these ones, so we wanted to emphasize the features that will be used or changed to fit our problem.

Let us start by providing the exact definitions.

4.1 Basic Definitions

Definition 1 *Let* $\mathbf{x} = [\underline{x}, \overline{x}]$ *be an interval.*

- *the value* $\Delta \stackrel{\text{def}}{=} (\overline{x} - \underline{x})/2$ *is called a* half-width *of this interval;*
- *the value* $\widetilde{x} \stackrel{\text{def}}{=} (\underline{x} + \overline{x})/2$ *is called a* midpoint *of this interval.*

Definition 2 *By an* interval data*, we mean a finite set of intervals* $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

Definition 3 *We say that the interval data* $\mathbf{x}_1, \ldots, \mathbf{x}_n$ satisfies the subset property *if* $[\underline{x}_i, \overline{x}_i] \not\subseteq (\underline{x}_j, \overline{x}_j)$ *for every* i *and* j.

4.2 Mean

Let us start our discussion with the simplest possible characteristic: the mean. The arithmetic average E is a monotonically increasing function of each of its n variables $x_1, \ldots, x_n$, so its smallest possible value $\underline{E}$ is attained when each value x_i is the smallest possible ($x_i = \underline{x}_i$) and its largest possible value is attained when $x_i = \overline{x}_i$ for all i. In other words, the range $\mathbf{E}$ of E is equal to $[E(\underline{x}_1, \ldots, x_n), E(\overline{x}_1, \ldots, \overline{x}_n)]$. In other words, $\underline{E} = \dfrac{1}{n}(\underline{x}_1 + \ldots + \underline{x}_n)$ and $\overline{E} = \dfrac{1}{n}(\overline{x}_1 + \ldots + \overline{x}_n)$.

4.3 Variance

4.3.1 Definitions and the Main Result

Definition 4 *For* n *real numbers* $x_1, \ldots, x_n$, *their* variance $V(x_1, \ldots, x_n)$ *is defined as*

$$V \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^2 = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - E^2,$$

where $E \stackrel{\text{def}}{=} \dfrac{1}{n} \cdot \sum_{i=1}^{n} x_i$.

Definition 5 *By the* interval variance $\mathbf{V}$ *of the interval data, we mean the interval*

$$\mathbf{V} \stackrel{\text{def}}{=} \{V(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

formed by the values $V(x_1, \ldots, x_n)$ *corresponding to different* $x_i \in \mathbf{x}_i$.

Theorem 1 *There exists an algorithm that computes the variance* $\mathbf{V}$ *of the interval data in time* $\mathcal{O}(n \cdot \log(n))$ *for all the cases in which the interval data satisfies the subset property.*

In order to compute the interval $\mathbf{V}$, we must compute both endpoints $\underline{V}$ and $\overline{V}$ of this interval. In [15], an algorithm is described that computes $\underline{V}$ in time $\mathcal{O}(n \cdot \log(n))$ for arbitrary interval data. So, to prove the main result, it is sufficient to describe a new algorithm for computing $\overline{V}$, and to prove that the new algorithm is correct and has the desired complexity.

4.3.2 Algorithm for Computing $\overline{V}$: Description

The algorithm for computing $\overline{V}$ is as follows:

- First, we sort n intervals $\mathbf{x}_i$ in lexicographic order:

$$\mathbf{x}_1 \leq_{\text{lex}} \mathbf{x}_2 \leq_{\text{lex}} \ldots \leq_{\text{lex}} \mathbf{x}_n,$$

  where $[\underline{a}, \overline{a}] \leq_{\text{lex}} [\underline{b}, \overline{b}]$ if and only if either $\underline{a} < \underline{b}$, or $\underline{a} = \underline{b}$ and $\overline{a} \leq \overline{b}$.

- Second, we use bisection to find the value k ($1 \leq k \leq n$) for which the following two inequalities hold:

$$\widetilde{x}_k + \frac{1}{n} \cdot \sum_{i=1}^{k-1} \Delta_i \leq \frac{1}{n} \cdot \sum_{i=k+1}^{n} \Delta_i + \frac{1}{n} \cdot \sum_{i=1}^{n} \widetilde{x}_i; \tag{4.1}$$

$$\widetilde{x}_{k+1} + \frac{1}{n} \cdot \sum_{i=1}^{k} \Delta_i \geq \frac{1}{n} \cdot \sum_{i=k+2}^{n} \Delta_i + \frac{1}{n} \cdot \sum_{i=1}^{n} \widetilde{x}_i. \tag{4.2}$$

  At each iteration of this bisection, we have an interval $[k^-, k^+]$ that is guaranteed to contain k. In the beginning, $k^- = 1$ and $k^+ = n$. At each stage, we compute the midpoint $k_{\text{mid}} = \lfloor (k^- + k^+)/2 \rfloor$, and check both inequalities (4.1) and (4.2) for $k = k_{\text{mid}}$. Then:

  - If both inequalities (4.1) and (4.2) hold for his k, this means that we have found the desired k.
  - If (4.1) holds but (4.2) does not hold, this means that the desired value k is larger than k_{mid}, so we keep k^+ and replace k^- with $k_{\text{mid}} + 1$.

– If (4.2) holds but (4.1) does not hold, this means that the desired value k is smaller than k_{mid}, so we keep k^- and replace k^+ with $k_{\text{mid}} - 1$.

- Once k is found, we compute

$$V_k \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{i=1}^{k} \underline{x}_i^2 + \frac{1}{n} \cdot \sum_{i=k+1}^{n} \overline{x}_i^2 - \left(\frac{1}{n} \cdot \sum_{i=1}^{k} \underline{x}_i + \frac{1}{n} \cdot \sum_{i=k+1}^{n} \overline{x}_i \right)^2. \tag{4.3}$$

This value is returned as the desired value $\overline{V}$.

4.3.3 Proof of Correctness and Complexity

Let us prove that this algorithm indeed produces the correct result and indeed requires time $\mathcal{O}(n \cdot \log(n))$.

1°. Let us first prove that if the interval data satisfies the subset property, then, after we sort these elements in lexicographic order, both the lower endpoints $\underline{x}_i$ and the upper endpoints $\overline{x}_i$ are sorted in non-decreasing order: $\underline{x}_i \le \underline{x}_{i+1}$ and $\overline{x}_i \le \overline{x}_{i+1}$.

Indeed, by definition of a lexicographic order, we always have $\underline{x}_i \le \underline{x}_{i+1}$. If $\underline{x}_i = \underline{x}_{i+1}$, then, by definition of the lexicographic order, we have $\overline{x}_i \le \overline{x}_{i+1}$.

If $\underline{x}_i < \underline{x}_{i+1}$, then we cannot have $\overline{x}_i > \overline{x}_{i+1}$ – otherwise, we would have $(\underline{x}_{i+1}, \overline{x}_{i+1}) \subset \mathbf{x}_i$ which would contradict the subset property. Hence, if $\underline{x}_i < \underline{x}_{i+1}$, we also have $\overline{x}_i \le \overline{x}_{i+1}$. The statement is proven.

It is known that sorting requires time $\mathcal{O}(n \cdot \log(n))$; see, e.g., [4].

In the following text, we will assume that the sequence of intervals has been sorted in this manner.

2°. Let us now prove that the desired maximum of the variance V is attained when each variable x_i is at one of the endpoints of the corresponding interval $\mathbf{x}_i$.

Let us prove this statement by reduction to a contradiction. Let us assume that the maximum is attained in the interior point of this interval. This would mean that in this

point, $\partial V/\partial x_i = 0$ and $\partial^2 V/\partial x_i^2 \le 0$. However, for the variance, we have $\partial V/\partial x_i = (2/n)\cdot(x_i - E)$, so the second derivative of the variance is equal to $\partial^2 V/\partial x_i^2 = (2/n)\cdot(1-1/n)$ and is, therefore, always positive. This contradiction proves that the maximum cannot be attained at an interior point. The statement is proven.

3°. Let us show the maximum is attained at a vector

$$x = (\underline{x}_1, \ldots, \underline{x}_k, \overline{x}_{k+1}, \ldots, \overline{x}_n) \tag{4.4}$$

in which we first have lower endpoints and then upper endpoints.

What we need to prove is that there exists a maximizing vector in which, once we have an upper endpoint, what follows will also be an upper endpoint, i.e., in which we cannot have $x_k = \overline{x}_k > \underline{x}_k$ and $x_{k+1} = \underline{x}_{k+1} < \overline{x}_{k+1}$.

For that, let us start with a maximizing vector in which this property does not hold, i.e., in which $x_k = \overline{x}_k > \underline{x}_k$ and $x_{k+1} = \underline{x}_{k+1} < \overline{x}_{k+1}$ for some k. Based on this vector, we will now construct a different maximizing vector with the desired property. For that, let us consider two cases: $\Delta_k < \Delta_{k+1}$ and $\Delta_k \ge \Delta_{k+1}$, where $\Delta_i \stackrel{\text{def}}{=} (\overline{x}_i - \underline{x}_i)/2$ is the half-width of the interval $\mathbf{x}_i$.

In the first case, let us replace $\overline{x}_k = \underline{x}_k + 2\Delta_k$ with $\underline{x}_k$, and $\underline{x}_{k+1}$ with $\underline{x}_{k+1} + 2\Delta_k$ (since $\Delta_k < \Delta_{k+1}$, this new value is $< \overline{x}_{k+1}$). Here, the average E remains the same, so the only difference between the new value V' of the variance and its old value V comes from the change in terms x_k^2 and x_{k+1}^2. In other words,

$$V' - V = \frac{1}{n}\cdot((\underline{x}_{k+1} + 2\Delta_k)^2 - \underline{x}_{k+1}^2) - \frac{1}{n}\cdot((\underline{x}_k + 2\Delta_k)^2 - \underline{x}_k^2).$$

Opening parentheses and simplifying the resulting expression, we conclude that $V' - V = (4\Delta_k/n)\cdot(\underline{x}_{k+1} - \underline{x}_k)$. Since V is the maximum, we must have $V' - V \le 0$, hence $\underline{x}_{k+1} \le \underline{x}_k$. Due to our ordering, we thus have $\underline{x}_{k+1} = \underline{x}_k$. Since we assumed that $\Delta_k < \Delta_{k+1}$, we have $\overline{x}_k = \underline{x}_k + 2\Delta_k < \overline{x}_{k+1} = \underline{x}_{k+1} + 2\Delta_{k+1}$, hence the interval $\mathbf{x}_k$ is a proper subset of $\mathbf{x}_{k+1}$ – which is impossible.

In the second case, when $\Delta_k \geq \Delta_{k+1}$, let us replace $\overline{x}_k$ with $\overline{x}_k - 2\Delta_{k+1}$ (which is still $\geq \underline{x}_k$), and $\underline{x}_{k+1} = \overline{x}_{k+1} - 2\Delta_{k+1}$ with $\overline{x}_{k+1}$. Here, the average E remains the same, and the only difference between the new value V' of the variance and its old value V comes from the change in terms x_k^2 and x_{k+1}^2, hence

$$V' - V = \frac{1}{n} \cdot (\overline{x}_{k+1}^2 - (\overline{x}_{k+1} - 2\Delta_{k+1})^2) - \frac{1}{n} \cdot (\overline{x}_k^2 - (\overline{x}_k - 2\Delta_{k+1})^2),$$

i.e., $V' - V = (4\Delta_{k+1}/n) \cdot (\overline{x}_{k+1} - \overline{x}_k)$. Since V is the maximum, we must have $V' - V \leq 0$, hence $\overline{x}_{k+1} \leq \overline{x}_k$. Due to our ordering, we thus have $\overline{x}_{k+1} = \overline{x}_k$. Since we assumed that $\Delta_k \geq \Delta_{k+1}$, we have $\underline{x}_k = \overline{x}_k - 2\Delta_k \geq \underline{x}_{k+1} = \overline{x}_{k+1} - 2\Delta_{k+1}$, i.e., $\mathbf{x}_k \subseteq \mathbf{x}_{k+1}$. Since intervals cannot be proper subsets of each other, we thus have $\mathbf{x}_k = \mathbf{x}_{k+1}$. In this case, we can simply swap the values x_k and x_{k+1}, variance will not change.

If necessary, we can perform this swap for all needed k; as a result, we get the maximizing vector with the desired property.

4°. Due to Part 3 of this proof, the desired value $\overline{V} = \max V$ is the largest of $n+1$ values (4.3) corresponding to $k = 0, 1, \ldots, n$.

In principle, to compute $\overline{V}$, we can therefore compute each of these values and find the largest of them. Computing each value takes $\mathcal{O}(n)$ times, so computing $n+1$ such values would require time $\mathcal{O}(n^2)$. Let us show that we can compute $\overline{V}$ faster.

We must find the index k for which V_k is the largest. For the desired k, we have $V_k \geq V_{k-1}$ and $V_k \geq V_{k+1}$. Due to (4.3), we conclude that

$$V_k - V_{k-1} = \frac{1}{n} \cdot \left(\underline{x}_k^2 - \overline{x}_k^2\right) -$$

$$\left(\frac{1}{n} \cdot \sum_{i=1}^{k} \underline{x}_i + \frac{1}{n} \cdot \sum_{i=k+1}^{n} \overline{x}_i\right)^2 + \left(\frac{1}{n} \cdot \sum_{i=1}^{k-1} \underline{x}_i + \frac{1}{n} \cdot \sum_{i=k}^{n} \overline{x}_i\right)^2. \tag{4.5}$$

Each pair of terms in the right-hand side of (4.5) can be simplified if we use the fact that $a^2 - b^2 = (a-b) \cdot (a+b)$ and use the notations Δ_k and $\widetilde{x}_k \stackrel{\text{def}}{=} (\underline{x}_k + \overline{x}_k)/2$. First, we get $\underline{x}_k^2 - \overline{x}_k^2 = (\underline{x}_k - \overline{x}_k) \cdot (\underline{x}_k + \overline{x}_k) = -4\Delta_k \cdot \widetilde{x}_k$. Second, we get

$$\left(\frac{1}{n} \cdot \sum_{i=1}^{k-1} \underline{x}_i + \frac{1}{n} \cdot \sum_{i=k}^{n} \overline{x}_i\right)^2 - \left(\frac{1}{n} \cdot \sum_{i=1}^{k} \underline{x}_i + \frac{1}{n} \cdot \sum_{i=k+1}^{n} \overline{x}_i\right)^2 =$$

$$\frac{2}{n}\cdot(\overline{x}_k-\underline{x}_k)\cdot\left(\frac{1}{n}\cdot\sum_{i=1}^{k-1}\underline{x}_i+\frac{1}{n}\cdot\widetilde{x}_k+\frac{1}{n}\cdot\sum_{i=k+1}^{n}\overline{x}_i\right).$$

Here, $\overline{x}_k-\underline{x}_k=2\Delta_k$, hence the formula (4.5) takes the following form:

$$V_k-V_{k-1}=\frac{4}{n}\cdot\Delta_k\cdot\left(-\widetilde{x}_k+\frac{1}{n}\cdot\sum_{i=1}^{k-1}\underline{x}_i+\frac{1}{n}\cdot\widetilde{x}_k+\frac{1}{n}\cdot\sum_{i=k+1}^{n}\overline{x}_i\right).$$

Since $V_k\ge V_{k-1}$ and $\Delta_k>0$, we conclude that

$$-\widetilde{x}_k+\frac{1}{n}\cdot\sum_{i=1}^{k-1}\underline{x}_i+\frac{1}{n}\cdot\widetilde{x}_k+\frac{1}{n}\cdot\sum_{i=k+1}^{n}\overline{x}_i\ge 0. \tag{4.6}$$

Substituting the expressions $\underline{x}_i=\widetilde{x}_i-\Delta_i$ and $\overline{x}_i=\widetilde{x}_i+\Delta_i$ into the formula (4.6) and moving all the negative terms to the other side of the inequality, we get the inequality (4.1). Similarly, the inequality $V_{k+1}\le V_k$ leads to (4.2).

When k increases, the left-hand side of the inequality (4.1) increases – because $\widetilde{x}_k$ increases as the average of the two increasing values $\underline{x}_k$ and $\overline{x}_k$, and the sum is increasing. Similarly, the right-hand side of this inequality decreases with k. Thus, if this inequality holds for k, it should also hold for all smaller values, i.e., for $k-1$, $k-2$, etc.

Similarly, in the second desired inequality (4.2), when k increases, the left-hand side of this inequality increases, while the right-hand side decreases. Thus, if this inequality is true for k, it is also true for $k+1$, $k+2$, ...

If both inequalities (4.1) and (4.2) are true for two different values $k<k'$, then they should both be true for all the values intermediate between k and k', i.e., for $k+1, k+2, \ldots, k'-1$. If (4.1) and and (4.2) are both true for k and $k+1$, this means that in both cases, we have equality, thus $V_k=V_{k+1}$, so it does not matter which of these values k we take.

Thus, modulo this equality case, there is, in effect, only one k for which both inequalities are true, and this k can be found by the bisection method as described in the above algorithm.

How long does this algorithm take? In the beginning, we only know that k belongs to the interval $[1,n]$ of width $\mathcal{O}(n)$. At each stage of the bisection step, we divide the interval

(containing k) in half. After I iterations, we decrease the width of this interval by a factor of 2^I. Thus, to find the exact value of k, we must have I for which $\mathcal{O}(n)/2^I = 1$, i.e., we need $I = \mathcal{O}(\log(n))$ iterations. On each iteration, we need $\mathcal{O}(n)$ steps, so we need a total of $\mathcal{O}(n \cdot \log(n))$ steps. With $\mathcal{O}(n \cdot \log(n))$ steps for sorting, and $\mathcal{O}(n)$ for computing the variance, we get a $\mathcal{O}(n \cdot \log(n))$ algorithm.

The theorem is proven.

4.4 Skewness

4.4.1 Definitions and the Main Result

Definition 6 *For n real numbers $x_1, \ldots, x_n$, their* skewness $S(x_1, \ldots, x_n)$ *is defined as*

$$S \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^3,$$

where $E \stackrel{\text{def}}{=} \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$.

For $n = 1$, we have $x_1 = E$, so the skewness is 0. For $n = 2$, we have $E = \frac{x_1 + x_2}{2}$, hence

$$S = \frac{1}{2} \cdot [((x_1 - E)^3 + (x_2 - E)^3] = \frac{1}{2} \cdot \left[\left(\frac{x_1 - x_2}{2} \right)^3 + \left(\frac{x_2 - x_1}{2} \right)^3 \right] = 0.$$

Thus, non-trivial skewness values are only possible for $n > 2$.

Definition 7 *By the* interval skewness $\mathbf{S}$ *of the interval data, we mean the interval*

$$\mathbf{S} \stackrel{\text{def}}{=} \{S(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}$$

formed by the values $S(x_1, \ldots, x_n)$ corresponding to different $x_i \in \mathbf{x}_i$.

Theorem 2 *There exists an algorithm that computes the skewness $\mathbf{S}$ of the interval data in time $\mathcal{O}(n^2)$ for all the cases in which the interval data satisfies the subset property.*

4.4.2 Proof of the Main Result

1°. In order to compute the interval $\mathbf{S}$, we must compute both endpoints $\underline{S}$ and $\overline{S}$ of this interval.

Let us first show that if we can compute $\overline{S}$, then we can easily compute $\underline{S}$ as well. Indeed, skewness is an odd function: $S(-x_1, \ldots, -x_n) = -S(x_1, \ldots, x_n)$; thus, for intervals $-\mathbf{x}_i = -[\underline{x}_i, \overline{x}_i] = [-\overline{x}_i, -\underline{x}_i]$, we have $\mathbf{S}(-\mathbf{x}_1, \ldots, -\mathbf{x}_n) = -\mathbf{S}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. From this relation between the skewness intervals, we can conclude that $\overline{S}(-\mathbf{x}_1, \ldots, -\mathbf{x}_n) = -\underline{S}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Thus, if we know how to compute $\overline{S}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ for an arbitrary collection of intervals $\mathbf{x}_i$, we can thus compute $\underline{S}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ as $-\overline{S}(-\mathbf{x}_1, \ldots, -\mathbf{x}_n)$

In view of this comment, in the remaining part of the proof, we will only consider an algorithm for computing $\overline{S}$.

2°. As we have shown while proving Theorem 1, since the interval data satisfies the subset property, after we sort these elements in lexicographic order, both the lower endpoints $\underline{x}_i$ and the upper endpoints $\overline{x}_i$ are sorted in non-decreasing order: $\underline{x}_i \le \underline{x}_{i+1}$ and $\overline{x}_i \le \overline{x}_{i+1}$.

3°. The maximum of a differentiable function $S(x_1, \ldots, x_n)$ on an interval $[\underline{x}_i, \overline{x}_i]$ can be attained either in an interior point of this interval, or at one of the endpoints.

If the maximum is attained at an interior point, then the first derivative is 0 $\left(\dfrac{\partial S}{\partial x_i} = 0\right)$ and the second derivative should be non-positive $\left(\dfrac{\partial^2 S}{\partial x_i^2} \le 0\right)$.

If the maximum is attained at the left endpoint, the function S cannot be increasing at this point, so we must have $\dfrac{\partial S}{\partial x_i} \le 0$. Similarly, if the maximum is attained at the right endpoint, the function S cannot be decreasing at this point, so we must have $\dfrac{\partial S}{\partial x_i} \ge 0$.

For skewness,

$$\frac{\partial S}{\partial x_i} = \frac{3}{n} \cdot (x_i - E)^2 - \frac{3}{n} \cdot \sum_{j=1}^{n} (x_j - E)^2 \cdot \frac{\partial E}{\partial x_i}.$$

Since $\dfrac{\partial E}{\partial x_i} = \dfrac{1}{n}$, we thus get $\dfrac{\partial S}{\partial x_i} = \dfrac{3}{n} \cdot ((x_i - E)^2 - V)$. So, the first derivative of S has the same sign as the expression $(x_i - E)^2 - V$.

To compute the second derivative of S, we must take into account that $\dfrac{\partial V}{\partial x_i} = \dfrac{2}{n}\cdot(x_i - E)$, hence

$$\frac{\partial^2 S}{\partial x_i^2} = \frac{3}{n}\cdot\left(2(x_i - E) - 2(x_i - E)\cdot\frac{1}{n} - \frac{2}{n}\cdot(x_i - E) + \frac{2}{n}\cdot\sum_{j=1}^{n}(x_j - E)\cdot\frac{1}{n}\right).$$

Since $\sum\limits_{j=1}^{n}(x_j - E) = 0$, we conclude that

$$\frac{\partial^2 S}{\partial x_i^2} = \frac{3}{n}\cdot 2\cdot\left(1 - \frac{2}{n}\right)\cdot(x_i - E).$$

We have already mentioned that the problem of computing skewness only makes sense for $n > 2$, because for $n \leq 2$, the skewness is identically 0. For $n > 2$, the second derivative $\dfrac{\partial^2 S}{\partial x_i^2}$ has the same sign as the expression $x_i - E$.

Thus, for skewness, we value $x_{\max} = (x_{1,\max}, \ldots, x_{i,\max}, \ldots, x_{n,\max})$ at which the maximum is attained satisfies one of the following three conditions:

- either $\underline{x}_i < x_{i,\max} < \overline{x}_i$, or $(x_{i,\max} - E_{\max})^2 - V_{\max} = 0$, and $x_{i,\max} - E_{\max} \leq 0$, where

$$E_{\max} = \frac{1}{n}\cdot\sum_{i=1}^{n} x_{i,\max}, \quad V_{\max} = \frac{1}{n}\cdot\sum_{i=1}^{n}(x_{i,\max} - E_{\max})^2;$$

- or $x_{i,\max} = \underline{x}_i$ and $(x_{i,\max} - E_{\max})^2 - V_{\max} \leq 0$,

- or $x_{i,\max} = \overline{x}_i$ and $(x_{i,\max} - E_{\max})^2 - V_{\max} \geq 0$.

In the first case, $(x_{i,\max} - E_{\max})^2 = V_{\max} = \sigma^2$, where we denoted $\sigma = \sqrt{V_{\max}}$, hence $x_{i,\max} - E_{\max} = \pm\sigma$. Since $x_{i,\max} - E_{\max} \leq 0$, we cannot have $x_{i,\max} - E_{\max} = \sigma$, so in this case, $x_{i,\max} = E_{\max} - \sigma$. In the second case, $(x_{i,\max} - E_{\max})^2 \leq V_{\max} = \sigma^2$, hence $E_{\max} - \sigma \leq x_{i,\max} \leq E_{\max} + \sigma$. In the third case, $(x_{i,\max} - E_{\max})^2 \geq V_{\max} = \sigma^2$, so either $x_{i,\max} \leq E_{\max} - \sigma$ or $x_{i,\max} \geq E_{\max} + \sigma$. So:

- either $\underline{x}_i < x_{i,\max} = E_{\max} - \sigma < \overline{x}_i$,

- or $x_{i,\max} = \underline{x}_i$ and $E_{\max} - \sigma \leq \underline{x}_i \leq E_{\max} + \sigma$,

- or $x_{i,\max} = \overline{x}_i$ and either $\overline{x}_i \le E_{\max} - \sigma$ or $\overline{x}_i \ge E_{\max} + \sigma$.

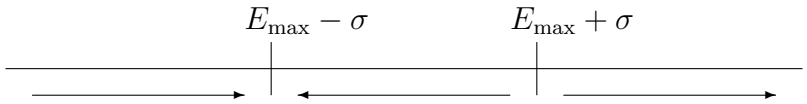
In all three cases, the desired maximum of the skewness S is attained when x_i is either at one of the endpoints of the corresponding interval $\mathbf{x}_i$, or has the value $\mu \stackrel{\text{def}}{=} E_{\max} - \sigma$.

4°. Let us now deduce more specific information about the values $x_{i,\max}$ at which the maximum is attained.

Based on the above description of possible cases, once we know how the intervals are located in relation to $E_{\max} - \sigma$ and $E_{\max} + \sigma$, we can sometimes uniquely determine the value $x_{i,\max}$ at which the maximum is attained. Namely,

- If $\overline{x}_i \le E_{\max} - \sigma$, then the maximum cannot be attained at an interior point and it cannot be attained at the value $\underline{x}_i$, so it is attained when $x_{i,\max} = \overline{x}_i$.
- If $\underline{x}_i \le E_{\max} - \sigma \le \overline{x}_i \le E_{\max} + \sigma$, then the maximum can only be attained when $x_{i,\max} = E_{\max} - \sigma$.
- If $E_{\max} - \sigma \le \underline{x}_i \le E_{\max} + \sigma$, then the maximum is attained at $x_{i,\max} = \underline{x}_i$.
- Finally, if $E_{\max} + \sigma \le \underline{x}_i$, then the maximum is attained at $x_{i,\max} = \overline{x}_i$.

These conclusions can be described in the following graphical manner, in which the arrows indicate the direction towards the corresponding maximum:



The only case when we cannot exactly determine the optimal value $x_{i,\max}$ is when the interval $\mathbf{x}_i$ contains the value $E_{\max} + \sigma$: it this case, we may have $x_{i,\max} = \overline{x}_i$, and we may also have $x_{i,\max} = \max(E_{\max} - \sigma, \underline{x}_i)$.

5°. Let us show that the maximum of skewness is always attained at a vector $x_{\max} = (x_{1,\max}, \ldots, x_{n,\max})$ which can be divided into three consecutive fragments (some of which may be empty):

- first, we have values $\overline{x}_i$ which are smaller than $E_{\max} - \sigma$;
- then, we have the values $\max(E_{\max} - \sigma, \underline{x}_i)$;
- finally, we have the values $\overline{x}_i$ which are larger than $E_{\max} + \sigma$.

All the intervals $\mathbf{x}_i$ that do not contain $E_{\max} + \sigma$ inside naturally fall into this scheme. The only intervals that we do need to consider to prove this result are the intervals that do contain $E_{\max} + \sigma$. For each of these intervals, the corresponding values $x_{i,\max}$ are either $\max(E_{\max} - \sigma, \underline{x}_i)$ or $\overline{x}_i$. What we claim is that after we sort the intervals in lexicographic order, we will first have the values equal to $\max(E_{\max} - \sigma, \underline{x}_i)$, and then the values equal to $\overline{x}_i$. In other words, once we have a value $x_{i,\max} = \overline{x}_i$, all the following values will also be of the same type.

We will show that if there is an optimizing vector $x_{\max}$ at which this condition is not satisfied, then we can rearrange it into a new vector with the same optimal value of S for which this condition holds.

Indeed, let us start with a vector for which, for some i, for two consequent intervals $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, a value $x_{i,\max} = \overline{x}_i \geq E_{\max} + \sigma$ is followed by a value

$$x_{i+1,\max} = \max(E_{\max} - \sigma, \underline{x}_{i+1}) \leq E_{\max} + \sigma.$$

If there are several such indices i, we take the smallest i with this property.

According to Part 2 of this proof, we have $\underline{x}_i \leq \underline{x}_{i+1} \leq E_{\max} + \sigma$ and $E_{\max} + \sigma \leq \overline{x}_i \leq \overline{x}_{i+1}$. Thus, $x_{i,\max} = \overline{x}_i \leq \overline{x}_{i+1}$ and $x_{i,\max} = \overline{x}_i \geq E_{\max} + \sigma \geq \underline{x}_{i+1}$; hence, $x_{i,\max} \in \mathbf{x}_{i+1}$. Similarly, $x_{i+1,\max} \in \mathbf{x}_i$. Thus, we can "swap" the values $x_{i,\max}$ and $x_{i+1,\max}$: as a new value of $x_{i,\max}$, we take the old value of $x_{i+1,\max}$, and vice versa. The swap does not change the average E and does not change the sample skewness S, so the function S attains the maximum at the new values as well.

As a result of this swap, if there is now a value i' for which $\overline{x}_{i'}$ is followed by

$$\max(E_{\max} - \sigma, \underline{x}_{i'+1}),$$

this value i' has to be equal to at least $i+1$. If there still is such an index i', we apply a new swap again and thus again increase the smallest problematic value i. After $\leq n$ such swaps, there will be no problematic cases anymore, so we will get a sequence which has the desired property.

6°. To determine the optimal vector $x_{\max}$, we must thus select a zone $[x_{(p)}, x_{(p+1)}]$ that contains $\mu = E_{\max} - \sigma$, and an index k at which the optimal value $x_{i,\max}$ switches from $\max(\mu, \underline{x}_i)$ to $\overline{x}_i$.

Once p and k are fixed, we can uniquely determine each of the optimal values $x_{i,\max}$ – some as known numbers, some as equal to the (unknown) value μ:

- when $\overline{x}_i \leq x_{(p)}$, we have $x_{i,\max} = \overline{x}_i$;
- when $\underline{x}_i < x_{(p)} < x_{(p+1)} \leq \overline{x}_i$ and $i < k$, we have $x_{i,\max} = \mu$;
- when $x_{(p+1)} \leq \underline{x}_i$ and $i < k$, we have $x_{i,\max} = \underline{x}_i$;
- finally, when $i \geq k$, we have $x_{i,\max} = \overline{x}_i$.

To find μ, we must use the fact that $\mu = E_{\max} - \sigma$. Specifically, the average $E_{\max}$ can be determined as

$$\frac{1}{n} \cdot \sum_{i \in N'} x_{i,\max} + \frac{n - n'}{n} \cdot (E_{\max} - \sigma) = E_{\max},$$

where the sum is taken over the set N' of all the indices for which $x_{i,\max}$ is known, and n' is the total number of such indices. Similarly, the sample second moment $E_{\max}^2 + \sigma^2$ can be determined as

$$\frac{1}{n} \cdot \sum_{i \in N'} x_{i,\max}^2 + \frac{n - n'}{n} \cdot (E_{\max} - \sigma)^2 = E_{\max}^2 + \sigma^2.$$

From the first of these equations, we can determine σ as a linear function of $E_{\max}$. Substituting this expression into the second equation, we get a quadratic equation with the only unknown σ, from which we can determine σ. Then, we can use the first equation to find $E_{\max}$ – and hence find $\mu = E_{\max} - \sigma$.

If the resulting value of μ is indeed within the zone $[x_{(p)}, x_{(p+1)}]$, then we compute the sample skewness for the corresponding values $x_{i,\max}$. Specifically, the skewness can be computed as

$$\frac{1}{n} \cdot \sum_{i=1}^{n} (x_{i,\max} - E_{\max})^3 =$$

$$\frac{1}{n} \cdot \sum_{i=1}^{n} x_{i,\max}^3 - \frac{3 \cdot E_{\max}}{n} \cdot \sum_{i=1}^{n} x_{i,\max}^2 + \frac{3 \cdot E_{\max}^2}{n} \cdot \sum_{i=1}^{n} x_{i,\max} - E_{\max}^3 =$$

$$\frac{1}{n} \cdot \sum_{i=1}^{n} x_{i,\max}^3 - \frac{3 \cdot E_{\max}}{n} \cdot \sum_{i=1}^{n} x_{i,\max}^2 + 2 \cdot E_{\max}^3 =$$

$$\frac{1}{n} \cdot \sum_{i \in N'} x_{i,\max}^3 + \frac{n - n'}{n} \cdot \mu^3 - \frac{3 \cdot E_{\max}}{n} \cdot \left(\sum_{i \in N'} x_{i,\max}^2 + (n - n') \cdot \mu^2 \right) + 2 \cdot E_{\max}^3.$$

The largest of these skewnesses is the desired value $\overline{S}$.

7°. How much times does this algorithm require? Sorting takes time $\mathcal{O}(n \cdot \log(n))$.

For n interval data points, we have $2n$ possible zone and n possible indices k – totally, $\mathcal{O}(n^2)$ possible pairs (p, k). For the first pair, computing the corresponding values n', $\sum_{i \in N'} x_{i,\max}$, $\sum_{i \in N'} x_{i,\max}^2$, and $\sum_{i \in N'} x_{i,\max}^3$ requires linear time. For each next pair, we, in general, change one value in comparison with the previous pair, so each new computation requires a constant number of steps. Thus, for $\mathcal{O}(n^2)$ pairs, we need $\mathcal{O}(n^2)$ time. (In some cases, we change more than one value, but still, each value changes only once, so we still need $\mathcal{O}(n^2)$ times.)

So, overall, we need time $\mathcal{O}(n \cdot \log(n)) + \mathcal{O}(n) + \mathcal{O}(n^2) = \mathcal{O}(n^2)$.

The theorem is proven.

Chapter 5

New Result: Computing Odd Moments under Privacy-Related Interval Uncertainty – The Last Piece of the Puzzle

As we have mentioned earlier, under the privacy-related interval uncertainty, there existed efficient algorithms for all standard statistical characteristics except for one – odd central moments M_{2p+1}. For such moments, the algorithm was only known for the skewness $S = M_3$. In this chapter, we describe a new efficient algorithm that covers this remaining case.

5.1 From Skewness to General Odd Moments: Similarities and Challenges

5.1.1 Problem: Reminder

In this chapter, we consider odd central moments

$$M_{2p+1} = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - E)^{2p+1}. \tag{5.1}$$

Our objective is, given n intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, to compute the range of the corresponding moment under interval uncertainty:

$$\mathbf{M}_{2p+1} \stackrel{\text{def}}{=} \{M_{2p+1}(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

5.1.2 Main Similarity: Every Odd Moment is an Odd Function

In general, computing an interval like $\mathbf{S} = [\underline{S}, \overline{S}]$ means computing its endpoints $\underline{S}$ and $\overline{S}$. For skewness, the problem was simplified by the fact that skewness $S(x_1, \ldots, x_n)$ is an odd function of its variables, i.e., that $S(-x_1, \ldots, -x_n) = -S(x_1, \ldots, x_n)$. This fact was used to reduce the problem of computing the lower endpoint $\underline{S}$ to computing the upper endpoint $\overline{S}$.

It turns out that this reduction does not use any specific feature of skewness, just the fact that skewness is an odd function, i.e., a function $f(x_1, \ldots, x_n)$ for which

$$f(-x_1, \ldots, -x_n) = -f(x_1, \ldots, x_n).$$

This fact enables us to attain a similar reduction for all odd-order moments M_{2p+1}.

Indeed, one can easily check that the odd moment M_{2p+1} is an odd function, i.e.,

$$M_{2p+1}(-x_1, -x_2, \ldots, -x_n) = -M_{2p+1}(x_1, x_2, \ldots, x_n).$$

Thus, for every intervals $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1], \ldots, \mathbf{x}_n = [\underline{x}_n, \overline{x}_n]$, we have

$$\mathbf{M}_{2p+1}(-\mathbf{x}_1, \ldots, -\mathbf{x}_n) = -\mathbf{M}_{2p+1}(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

For an arbitrary interval $\mathbf{a} = [\underline{a}, \overline{a}]$, we have

$$-\mathbf{a} = -[\underline{a}, \overline{a}] = [-\overline{a}, -\underline{a}].$$

In particular, we have

$$-\mathbf{x}_i = -[\underline{x}_i, \overline{x}_i] = [-\overline{x}_i, -\underline{x}_i],$$

and

$$-\mathbf{M}_{2p+1} = -[\underline{M}_{2p+1}, \overline{M}_{2p+1}] = [-\overline{M}_{2p+1}, -\underline{M}_{2p+1}].$$

Thus, we can conclude that

$$\overline{M}_{2p+1}(-\mathbf{x}_1, -\mathbf{x}_2, \ldots, -\mathbf{x}_n) = -\underline{M}_{2p+1}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n),$$

and thus,

$$\underline{M}_{2p+1}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = -\overline{M}_{2p+1}(-\mathbf{x}_1, -\mathbf{x}_2, \ldots, -\mathbf{x}_n).$$

Therefore, once we know how to compute the upper endpoint $\overline{M}_{2p+1}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ for arbitrary privacy-related intervals, we will then be able to compute the corresponding lower endpoint $\underline{M}_{2p+1}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ by using the above formula.

So, just like for skewness, it is sufficient to know how to compute the upper endpoint $\overline{M}_{2p+1}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$. In other words, we need to compute the maximum of the odd central moment $M_{2p+1}(x_1, \ldots, x_n)$ when $x_i \in \mathbf{x}_i$.

5.1.3 Main Challenge: Computing the Maximum of a General Odd Moment is Much More Difficult than for Skewness

The justification of the above algorithm for skewness is based on a well-known fact that when a function $f(x_1, \ldots, x_n)$ attains its maximum at an interval point $x_i \in (\underline{x}_i, \overline{x}_i)$, then its corresponding partial derivative $\dfrac{\partial f}{\partial x_i}$ is equal to 0:

$$\frac{\partial f}{\partial x_i} = 0.$$

For the central odd moment,

$$\frac{\partial M_{2p+1}}{\partial x_i} = \frac{2p+1}{n} \cdot \left((x_i - E)^{2p} - \frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E)^{2p} \right),$$

and thus, the conditions that the derivative is equal to 0 is equivalent to

$$\frac{\partial M_{2p+1}}{\partial x_i} = \frac{2p+1}{n} \cdot \left((x_i - E)^{2p} - \frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E)^{2p} \right) = 0. \tag{5.2}$$

The justification of the skewness algorithm also took into account that not all points at which the first derivative is 0 are maxima: at a point at which the function attains is minimum, the derivative is also equal to 0. Internal points at which the maximum of the

function $f(x_1, \ldots, x_n)$ is attained have an additional property that the second derivative should be non-positive:

$$\frac{\partial^2 f}{\partial x_i^2} \le 0.$$

For the odd moment M_{2p+1}, we get

$$\frac{\partial^2 M_{2p+1}}{\partial x_i^2} = \frac{(2p+1)\cdot(2p)}{n^2} \cdot \left((n-2)\cdot(x_i - E)^{2p-1} + \frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E)^{2p-1} \right). \tag{5.3}$$

Thus, at a point where the maximum is attained, we must have

$$\frac{\partial^2 M_{2p+1}}{\partial x_i^2} = \frac{(2p+1)\cdot(2p)}{n^2} \cdot \left((n-2)\cdot(x_i - E)^{2p-1} + \frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E)^{2p-1} \right) \le 0.$$

In the particular case of the third central moment – the skewness, when $p = 1$ – the expression for the second derivative can be simplified. Indeed, in this case,

$$\frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E)^{2p-1} = \frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E) = \frac{1}{n} \cdot \sum_{j=1}^{n} x_j - \frac{1}{n} \cdot n \cdot E = E - E = 0.$$

Therefore, in this specific case the sign of the second derivative (5.3) is the same as the sign of the expression $(n-2)\cdot(x_i - E)$.

We have mentioned that for $n \le 2$, the skewness is always equal to 0, so the problem of computing skewness only starts with $n > 2$. For such n, the sign of the above expression coincides with the sign of the difference $x_i - E$. This observation leads to the algorithm described in the previous chapter.

However, in the general case $p > 1$, we can no longer simplify the expression (5.3) for the second derivative. Thus, we can no longer easily exploit the second-derivative necessary criterion for the maximum.

5.2 Overcoming the Challenge: Our New Approach

5.2.1 First Derivative-Based Criteria for Maximum: General Case

Since we cannot easily use the second derivative to locate the values at which the desired function M_{2p+1} attains its maximum, we will have to restrict ourselves only to the criteria

based on the first derivative.

These criteria are based on the known formulas of calculus. It is well known that if a differentiable function $f(x)$ is defined for all real numbers and attains its global maximum at some point $x_{\max}$, then its derivative $\dfrac{df}{dx}$ is equal to 0 at this point $x_{\max}$.

A similar result holds for an arbitrary function $f(x)$ which is defined on an interval $[\underline{x}, \overline{x}]$ and which is differentiable everywhere on this interval – in the sense that the limit

$$\frac{df}{dx}(x) \stackrel{\text{def}}{=} \lim_{h \to 0,\ x+h \in [\underline{x}, \overline{x}]} \frac{f(x+h) - f(x)}{h}$$

is defined for every $x \in [\underline{x}, \overline{x}]$. If such a function $f(x)$ attains its maximum on the interval $[\underline{x}, \overline{x}]$ at some point $x_{\max} \in [\underline{x}, \overline{x}]$, then:

- if the maximum is attained at an inside point $x_{\max} \in (\underline{x}, \overline{x})$, then $\dfrac{df}{dx}(x_{\max}) = 0$;
- if the maximum is attained at the left endpoint $x_{\max} = \underline{x}$, then $\dfrac{df}{dx}(x_{\max}) \le 0$;
- finally, if the maximum is attained at the right endpoint $x_{\max} = \overline{x}$, then $\dfrac{df}{dx}(x_{\max}) \ge 0$.

This argument can be naturally generalized to a differentiable function $f(x_1, \ldots, x_n)$ of several variables defined on a box

$$[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n].$$

Indeed, when a function of several variables attains its maximum at some point

$$x_{\max} = (x_{1,\max}, \ldots, x_{n,\max}),$$

this means that for every i, the corresponding function of one variable

$$f_i(x_i) \stackrel{\text{def}}{=} f(x_{1,\max}, \ldots, x_{i-1,\max}, x_i, x_{i+1,\max}, \ldots, x_{n,\max})$$

also attains its maximum at the point $x_i = x_{i,\max}$. Thus, we can formulate the criteria for this maximum in terms of the derivatives of this auxiliary functions $f_i(x_i)$ – and these derivatives are partial derivatives of the original function $f(x_1, \ldots, x_n)$.

So, we arrive at the following conditions. If a differentiable function $f(x_1, \ldots, x_n)$ attains its maximum on the intervals $[\underline{x}_1, \overline{x}_1], \ldots, [\underline{x}_n, \overline{x}_n]$, at some point $x_{\max} = (x_{1,\max}, \ldots, x_{n,\max})$, for which $x_{i,\max} \in [\underline{x}_i, \overline{x}_i]$, then, for every i:

- if the maximum is attained at an inside point $x_{i,\max} \in (\underline{x}_i, \overline{x}_i)$, then $\dfrac{\partial f}{\partial x_i}(x_{\max}) = 0$;
- if the maximum is attained at the left endpoint $x_{i,\max} = \underline{x}_i$, then $\dfrac{\partial f}{\partial x_i}(x_{\max}) \le 0$;
- finally, if the maximum is attained at the right endpoint $x_{i,\max} = \overline{x}_i$, then $\dfrac{\partial f}{\partial x_i}(x_{\max}) \ge 0$.

5.2.2 First Derivative-Based Criteria for Maximum: Case of Odd Moments

For our function M_{2p+1}, the partial derivative is determined by the formula (5.2). Thus, for every point $x = (x_1, \ldots, x_n)$, the sign of the corresponding first derivative coincides with the sign of the difference

$$(x_i - E)^{2p} - \frac{1}{n} \cdot \sum_{j=1}^{n} (x_j - E)^{2p}.$$

Therefore:

- if $\dfrac{\partial M_{2p+1}}{\partial x_i} \ge 0$, then $(x_i - E)^{2p} \ge \dfrac{1}{n} \sum_{j=1}^{n} (x_j - E)^{2p}$;
- if $\dfrac{\partial M_{2p+1}}{\partial x_i} \le 0$, then $(x_i - E)^{2p} \le \dfrac{1}{n} \sum_{j=1}^{n} (x_j - E)^{2p}$;
- if $\dfrac{\partial M_{2p+1}}{\partial x_i} = 0$, then $(x_i - E)^{2p} = \dfrac{1}{n} \sum_{j=1}^{n} (x_j - E)^{2p}$.

So, at the point $x_{\max} = (x_{1,\max}, \ldots, x_{n,max})$ at which the maximum is attained, the following conditions are satisfied for every i:

- if $x_{i,\max} = \overline{x}_i$ then $(x_{i,\max} - E_{\max})^{2p} \ge \dfrac{1}{n} \sum_{j=1}^{n} (x_{j,\max} - E_{\max})^{2p}$;

- if $x_{i,\max} = \underline{x}_i$, then $(x_{i,\max} - E_{\max})^{2p} \le \dfrac{1}{n}\sum_{j=1}^{n}(x_{j,\max} - E_{\max})^{2p}$;
- if $\underline{x}_i < x_{i,\max} < \overline{x}_i$, then $(x_{i,\max} - E_{\max})^{2p} = \dfrac{1}{n}\sum_{j=1}^{n}(x_{j,\max} - E_{\max})^{2p}$.

Here, $E_{\max} \stackrel{\text{def}}{=} \dfrac{1}{n} \cdot \sum_{j=1}^{n} x_{j,\max}$.

5.2.3 First Derivative-Based Criterion for Maximum: Simplification

To simplify our notation, let us denote

$$\alpha \stackrel{\text{def}}{=} \sqrt[2p]{\frac{1}{n} \cdot \sum_{j=1}^{n}(x_{j,\max} - E_{\max})^{2p}}.$$

Taking the $2p^{th}$ root in both sides of the above inequalities, we obtain the following simplified conditions:

- if $x_{i,\max} = \overline{x}_i$ then $|x_{i,\max} - E_{\max}| \ge \alpha$;
- if $x_{i,\max} = \underline{x}_i$, then $|x_{i,\max} - E_{\max}| \le \alpha$;
- if $\underline{x}_i < x_{i,\max} < \overline{x}_i$, then $|x_{i,\max} - E_{\max}| = \alpha$.

In all these cases, the maximum is attained when x_i is equal to one of the four values $\underline{x}_i$, $\overline{x}_i$, $E_{\max} - \alpha$, or $E_{\max} + \alpha$.

5.2.4 Locations of Intervals Relative to "Threshold" Values: Possible Cases

The condition $|x_{i,\max} - E_{\max}| \le \alpha$ means that $E_{\max} - \alpha \le x_{i,\max} \le E_{\max} + \alpha$. Similarly, the condition $|x_{i,\max} - E_{\max}| \ge \alpha$ means that either $x_{i,\max} \le E_{\max} - \alpha$ or $x_{i,\max} \ge E_{\max} + \alpha$. In both cases, what is important is the relation between $x_{i,\max}$ and the two values $E_{\max} - \alpha$

and $E_{\max} + \alpha$. Let us therefore consider all possible locations of the endpoints $\underline{x}_i$ and $\overline{x}_i$ in relation to these "threshold" values.

In comparison to these values $E_{\max} - \alpha$ and $E_{\max} + \alpha$, each of the endpoints $x_i = \underline{x}_i$ and $x_i = \overline{x}_i$ can be in one of three positions:

- it can be smaller than $E_{\max} - \alpha$: $x_i < E_{\max} - \alpha$;
- it can be in between $E_{\max} - \alpha$ and $E_{\max} + \alpha$: $E_{\max} - \alpha \le x_i \le E_{\max} + \alpha$; and
- it can be larger than $E_{\max} + \alpha$, i.e., $x_i > E_{\max} + \alpha$.

In particular, the upper endpoint $\overline{x}_i$ can be in one of these three positions. Then:

- if the upper endpoint is smaller than $E_{\max} - \alpha$, then the lower endpoint $\underline{x}_i \le \overline{x}_i$ is also smaller than $E_{\max} - \alpha$;
- if the upper endpoint is in between $E_{\max} - \alpha$ and $E_{\max} + \alpha$, then the lower endpoint $\underline{x}_i \le \overline{x}_i$ can be:
  - either smaller than $E_{\max} - \alpha$,
  - or also in between $E_{\max} - \alpha$ and $E_{\max} + \alpha$;
- finally, if the upper endpoint is larger than $E_{\max} + \alpha$, then for the lower endpoint $\underline{x}_i \le \overline{x}_i$, all three positions are possible:
  - it can be smaller than $E_{\max} - \alpha$: $\underline{x}_i < E_{\max} - \alpha$;
  - it can be in between $E_{\max} - \alpha$ and $E_{\max} + \alpha$: $E_{\max} - \alpha \le \underline{x}_i \le E_{\max} + \alpha$; and
  - it can be larger than $E_{\max} + \alpha$, i.e., $\underline{x}_i > E_{\max} + \alpha$.

Thus, with respect to the location of a specific pair of endpoints $\underline{x}_i \le \overline{x}_i$, we have the following possibilities:

1. we can have $\underline{x}_i \le \overline{x}_i < E_{\max} - \alpha$;

2. we can have $\underline{x}_i < E_{\max} - \alpha \le \overline{x}_i \le E_{\max} + \alpha$;

3. we can have $E_{\max} - \alpha \le \underline{x}_i \le \overline{x}_i \le E_{\max} + \alpha$;

4. we can have $\underline{x}_i < E_{\max} - \alpha < E_{\max} + \alpha < \overline{x}_i$;

5. we can have $E_{\max} - \alpha \le \underline{x}_i \le E_{\max} + \alpha < \overline{x}_i$, and

6. we can have $E_{\max} + \alpha < \underline{x}_i \le \overline{x}_i$.

5.2.5 Where Can the Maximum Be Attained: Case by Case Analysis

In the previous subsection, we described possible locations of the intervals relative to the threshold values $E_{\max} - \alpha$ and $E_{\max} + \alpha$. Let us analyze these locations one by one, and analyze where for each location, the maximum can be attained.

5.2.6 First Case

In the first case, for all $x_i \in [\underline{x}_i, \overline{x}_i]$, we have $x_i \le \overline{x}_i < E_{\max} - \alpha$ and therefore, $x_{i,\max} < E_{\max} - \alpha$. Thus, we have $|x_{i,\max} - E_{\max}| > \alpha$. So, in this case,

- we cannot have $x_{i,\max} = \underline{x}_i$, because then we would have $|x_{i,\max} - E_{\max}| \le \alpha$;
- similarly, we cannot have $\underline{x}_i < x_{i,\max} < \overline{x}_i$, because then we would have

$$|x_{i,\max} - E_{\max}| = \alpha.$$

Thus, the only remaining possibility is $x_{i,\max} = \overline{x}_i$. So, in the first case, the maximum of the odd moment is attained when $x_{i,\max} = \overline{x}_i$.

5.2.7 Second Case

Let us now consider the second case when $\underline{x}_i < E_{\max} - \alpha \le \overline{x}_i \le E_{\max} + \alpha$.

In this case, the maximum cannot be attained at the point $\underline{x}_i$, because then we would have $|\underline{x}_i - E_{\max}| \le \alpha$, but we have $\underline{x}_i < E_{\max} - \alpha$ and hence $|\underline{x}_i - E_{\max}| > \alpha$.

If the maximum is attained at the point $\overline{x}_i$, then we should have $|\overline{x}_i - E_{\max}| \ge \alpha$, i.e., either $\overline{x}_i \le E_{\max} - \alpha$ or $\overline{x}_i \ge E_{\max} + \alpha$. Since $\underline{x}_i \le E_{\max} + \alpha \le \overline{x}_i$, the only two possibilities is when $\overline{x}_i = E_{\max} - \alpha$ or when $\overline{x}_i = E_{\max} + \alpha$.

If the maximum is attained at an interior point of the interval $[\underline{x}_i, \overline{x}_i]$, then it is attained at one of the points $E_{\max} - \alpha$ or $E_{\max} + \alpha$. The only case when $E_{\max} + \alpha$ is in our interval $[\underline{x}_i, \overline{x}_i]$ is when $\overline{x}_i = E_{\max} + \alpha$.

So, in the second case, the maximum is attained either at the point $E_{\max} - \alpha$ or (if $\overline{x}_i = E_{\max} + \alpha$) at the point $E_{\max} + \alpha$.

5.2.8 Third Case

Let us consider the third case when $E_{\max} - \alpha \le \underline{x}_i \le \overline{x}_i \le E_{\max} + \alpha$, i.e., when $[\underline{x}_i, \overline{x}_i] \subseteq [E_{\max} - \alpha, E_{\max} + \alpha]$.

In this case, if the maximum to be attained at the value $\overline{x}_i$, then we should have $x_i \le E_{\max} - \alpha$ or $x_i \ge E_{\max} + \alpha$. Since $[\underline{x}_i, \overline{x}_i] \subseteq [E_{\max} - \alpha, E_{\max} + \alpha]$, the only possibility for these inequalities to hold is when either $\underline{x}_i = E_{\max} - \alpha$ or $\overline{x}_i = E_{\max} + \alpha$. In other words, the only possibility for the maximum to be attained at the value $\overline{x}_i$ if when this value is equal to $E_{\max} + \alpha$.

If the maximum cannot be attained for $x_i \in (\underline{x}_i, \overline{x}_i)$, then we should have $x_{i,\max} = E_{\max} - \alpha$ or $x_{i,\max} = E_{\max} + \alpha$. Since the whole interval $[\underline{x}_i, \overline{x}_i]$ is located inside the interval $[E_{\max} - \alpha, E_{\max} + \alpha]$, neither $E_{\max} - \alpha$ not $E_{\max} + \alpha$ can be an interior point of the interval $[\underline{x}_i, \overline{x}_i]$. So, this case is not possible.

So, in the third case, the maximum is attained either at the lower endpoint $\underline{x}_i$, or – if the interval $[\underline{x}_i, \overline{x}_i]$ ends at exactly the point $\overline{x}_i$ – at the point $\overline{x}_i$.

5.2.9 Fourth Case

Let us consider the fourth case when $\underline{x}_i < E_{\max} - \alpha < E_{\max} + \alpha < \overline{x}_i$.

In this case, the maximum cannot be attained at the lower endpoint $\underline{x}_i$, because then we should have $|\underline{x}_i - E_{\max}| \le \alpha$ for this endpoint, and we actually have $\underline{x}_i < E_{\max} - \alpha$ hence $|\underline{x}_i - E_{\max}| > \alpha$.

Thus, in the fourth case, we only have 3 possibilities: the maximum is attained at $E_{\max} - \alpha$, at $E_{\max} + \alpha$, or at $\overline{x}_i$.

5.2.10 Fifth Case

Let us consider the fifth case when $E_{\max} - \alpha \le \underline{x}_i \le E_{\max} + \alpha < \overline{x}_i$.

In general, the maximum is attained either at $\underline{x}_i$, or at $\overline{x}_i$, or at $E_{\max} - \alpha$, or at $E_{\max} + \alpha$. Since $\underline{x}_i \ge E_{\max} - \alpha$, the only possibility for $E_{\max} - \alpha$ to be within the interval $[\underline{x}_i, \overline{x}_i]$ is when $E_{\max} - \alpha = \underline{x}_i$. Thus, in the case when $x_{i,\max} = E_{\max} - \alpha$, we also have $x_{i,\max} = \underline{x}_i$.

So, in this case, we must consider three possibilities: the maximum is attained at $\underline{x}_i$, at $E_{\max} + \alpha$, or at $\overline{x}_i$.

5.2.11 Sixth Case

In the sixth case, for all $x_i \in [\underline{x}_i, \overline{x}_i]$, we have $x_i \ge \underline{x}_i > E_{\max} + \alpha$ and therefore, $x_{i,\max} > E_{\max} + \alpha$. Thus, we have $|x_{i,\max} - E_{\max}| > \alpha$. So, in this case, similarly to the first case:

- we cannot have $x_{i,\max} = \underline{x}_i$, because then we would have $|x_{i,\max} - E_{\max}| \le \alpha$;
- similarly, we cannot have $\underline{x}_i < x_{i,\max} < \overline{x}_i$, because then we would have

$$|x_{i,\max} - E_{\max}| = \alpha.$$

Thus, the only remaining possibility is $x_{i,\max} = \overline{x}_i$. So, in the sixth case, the maximum of the odd moment is attained when $x_{i,\max} = \overline{x}_i$.

5.2.12 Summary of the Six Cases

For every interval $[\underline{x}_i, \overline{x}_i]$ which does not contain $E_{\max} + \alpha$, the value where the maximum is attained is uniquely determined:

- if $\overline{x}_i < E_{\max} - \alpha$, then the maximum is attained at the value $\overline{x}_i$;
- if $\underline{x}_i < E_{\max} - \alpha \leq \overline{x}_i < E_{\max} + \alpha$, then the maximum is attained at the value $E_{\max} - \alpha$;
- if $E_{\max} - \alpha \leq \underline{x}_i \leq \overline{x}_i < E_{\max} + \alpha$, then the maximum is attained at the value $\underline{x}_i$;
- if $E_{\max} + \alpha < \underline{x}_i \leq \overline{x}_i$, then the maximum is attained at the value $\overline{x}_i$.

For intervals which contain $E_{\max} + \alpha$, the maximum can be attained either at this point or at several other points:

- if $\underline{x}_i < E_{\max} - \alpha \leq \overline{x}_i = E_{\max} + \alpha$, then the maximum is attained either at $E_{\max} - \alpha$ or at $E_{\max} + \alpha$;
- if $E_{\max} - \alpha \leq \underline{x}_i \leq \overline{x}_i = E_{\max} + \alpha$, then the maximum is attained either at $\underline{x}_i$ or at $E_{\max} + \alpha$;
- if $\underline{x}_i < E_{\max} - \alpha < E_{\max} + \alpha < \overline{x}_i$, then the maximum is attained either at $E_{\max} - \alpha$ or at $E_{\max} + \alpha$ or at $\overline{x}_i$;
- $E_{\max} - \alpha \leq \underline{x}_i \leq E_{\max} + \alpha < \overline{x}_i$, then the maximum is attained either at $\underline{x}_i$ or at $E_{\max} + \alpha$ or at $\overline{x}_i$.

5.3 General Exponential Time Algorithm for Computing the Odd Moments

5.3.1 Main Idea: Considering all 4^n Possibilities

Before we describe an efficient algorithm for the privacy case, let us describe a general-case exponential-time algorithm.

Our analysis shows that, in general, for each of n intervals $[\underline{x}_i, \overline{x}_i]$, the maximum of the odd moment can be attained at one of the four values $\underline{x}_i$, $\overline{x}_i$, $E_{\max} - \alpha$, and $E_{\max} + \alpha$. For each value i, we have four possibilities. Thus, in general, we have 4^n possibilities.

Each possibility can be described by describing the corresponding four sets:

- the set $\underline{I}$ of all the indices i for which the maximum is attained at the value $\underline{x}_i$;
- the set $\overline{I}$ of all the indices i for which the maximum is attained at the value $\overline{x}_i$;
- the set I^- of all the indices i for which the maximum is attained at the value $E_{\max} - \alpha$; and
- the set I^+ of all the indices i for which the maximum is attained at the value $E_{\max} + \alpha$.

Once we know $E_{\max}$ and α, we can explicitly compute the corresponding odd central moment. Then, we can find the desired largest possible value of this moment by taking the largest of these 4^n values. The remaining question is how we can compute $E_{\max}$ and α.

5.3.2 How to Compute $E_{\max}$ and α for Each Possibility

To find $E_{\max}$ and α, we will use the definitions of $E_{\max}$ and α:

- $E_{\max}$ is the arithmetic average of the selected values $x_{i,\max}$:

$$E_{\max} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_{i,\max},$$

hence

$$n \cdot E_{\max} = \sum_{i=1}^{n} x_{i,\max};$$

- α^{2p} is the $(2p)$-th central moment of these values:

$$M_{2p} = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_{i,\max} - E_{\max})^{2p} = \alpha^{2p}$$

hence

$$\sum_{i=1}^{n} (x_{i,\max} - E_{\max})^{2p} = n \cdot \alpha^{2p}.$$

Let us first use the definition of $E_{\max}$.

Let n^- denote the number of elements in the set I^-, and let n^+ denote the number of elements in the set I^+. In these terms, the fact that $E_{\max}$ is the arithmetic average of all the values $x_{i,\max}$ can be rewritten as

$$n \cdot E_{\max} = \sum_{i \in \underline{I}} \underline{x}_i + \sum_{i \in \overline{I}} \overline{x}_i + n^- \cdot (E_{\max} - \alpha) + n^+ \cdot (E_{\max} + \alpha).$$

By moving all the terms containing $E_{\max}$ into the left-hand side, we conclude that

$$(n - n^- - n^+) \cdot E_{\max} = \sum_{i \in \underline{I}} \underline{x}_i + \sum_{i \in \overline{I}} \overline{x}_i + (n^+ - n^-) \cdot \alpha.$$

In the non-degenerate case when at least one of the sets $\underline{I}$ and $\overline{I}$ is non-empty, we have $n^- + n^+ < n$ and thus, from the above equation, we can deduce an explicit expression for $E_{\max}$ in terms of α:

$$E_{\max} = \frac{1}{n - n^- - n^+} \cdot \sum_{i \in \underline{I}} \underline{x}_i + \frac{1}{n - n^- - n^+} \cdot \sum_{i \in \overline{I}} \overline{x}_i + \frac{n^+ - n^-}{n - n^- - n^+} \cdot \alpha.$$

(We will describe the degenerate case later).

By definition of α, we have

$$\sum_{i=1}^{n} (x_{i,\max} - E_{\max})^{2p} = n \cdot \alpha^{2p}.$$

Dividing indices i into 4 subsets, we conclude that

$$\sum_{i\in\underline{I}}(\underline{x}_i - E_{\max})^{2p} + \sum_{i\in\overline{I}}(\overline{x}_i - E_{\max})^{2p} + (n^- + n^+)\cdot\alpha^{2p} = n\cdot\alpha^{2p}.$$

By moving the terms containing α^{2p} into the right hand side, we conclude that

$$\sum_{i\in\underline{I}}(\underline{x}_i - E_{\max})^{2p} + \sum_{i\in\overline{I}}(\overline{x}_i - E_{\max})^{2p} = (n - n^- - n^+)\cdot\alpha^{2p}.$$

Substituting the above linear expression for $E_{\max}$ in terms of α, we get an equation of order $2p$ with a single unknown α. There are known and efficient methods for finding all $\le 2p$ roots of this equation, such as Newtons' method or bisection; see, e.g., [1]. For each p, the computation time does not depend on the number n of inputs and is, thus, $\mathcal{O}(1)$.

Once we have found α, we can then find $E_{\max}$, and check that the corresponding selections are within the given intervals, i.e., that

- for all $i \in I^-$, we have $\underline{x}_i \le E_{\max} - \alpha \le \overline{x}_i$, and
- for all $i \in I^+$, we have $\underline{x}_i \le E_{\max} + \alpha \le \overline{x}_i$.

Once we checked that, we can then compute the corresponding value of the central moment

$$M_{2p+1} = \frac{1}{n}\cdot\sum_{i=1}^{n}(x_{i,\max} - E_{\max})^{2p+1}.$$

The largest of the values M_{2p+1} corresponding to all the possibilities is the desired maximum.

Since we have 4^n possibilities, we thus get an exponential-time algorithm.

5.3.3 Degenerate Case

We are almost ready to describe the resulting general algorithm, we only need to show how to compute $E_{\max}$ and α in the degenerate case, when $\underline{I} = \overline{I} = \emptyset$ and $n^- + n^+ = n$.

If $n^- + n^+ = n$, i.e., if $\underline{I} = \overline{I} = \emptyset$, then the above condition that $E_{\max}$ is the average means that $n^- = n^+$ and hence $n^- = n^+ = n/2$. In this case,

- half of the values $x_{i,\max}$ are equal to $x_{i,\max} = E_{\max} - \alpha$ and,
- half to $x_{i,\max} = E_{\max} + \alpha$.

Thus, for half of the values i, $x_{i,\max} - E_{\max} = \alpha$ and for the remaining half, $x_{i,\max} - E_{\max} = -\alpha$. So, for half of the indices i, we have $(x_{i,\max} - E_{\max}) = \alpha^{2p+1}$ and for the remaining half, we have $(x_{i,\max} - E_{\max})^{2p+1} = (-\alpha)^{2p+1} = -\alpha^{2p+1}$. Therefore, we have

$$M_{2p+1} = \frac{n}{2} \cdot \alpha^{2p+1} + \frac{n}{2} \cdot (-\alpha^{2p+1}) = 0.$$

So, in all such degenerate case, the odd central moment is equal to 0.

The only remaining question is how, given n intervals $[\underline{x}_i, \overline{x}_i]$, we can check whether such values exist, i.e., where it is possible to have two value $E_{\max} - \alpha$ and $E_{\max} + \alpha$ such that half of the interval contain the first value and half of the intervals contain the second value.

Let us consider the half of the indices i for which the corresponding intervals contain $E_{\max} - \alpha$, i.e., $\underline{x}_i \le E_{\max} - \alpha \le \overline{E}_i$. This means that the value $E_{\max} - \alpha$ is between the largest of the corresponding lower endpoints $\underline{x}_i$ and the smallest of the corresponding endpoints $\overline{x}_i$. In principle, we can therefore take the largest of these lower endpoints $\underline{x}_i$ instead of $E_{\max} - \alpha$. Thus, we can take

$$E_{\max} - \alpha = \max_{i \in I^-} \underline{x}_i$$

and similarly,

$$E_{\max} + \alpha = \max_{i \in I^+} \underline{x}_i.$$

Now, we are ready to describe the general algorithm.

5.3.4 Resulting Algorithm

We consider all 4^n subdivisions of the set $\{1, \ldots, n\}$ of all the indices $i \le n$ into 4 disjoint subsets $\underline{I}$, $\overline{I}$, I^-, and I^+. For each subdivision for which $\underline{I} \ne \emptyset$ or $\overline{I} \ne \emptyset$, we do the following:

- First, we find the coefficients of the linear expression of the unknown value $E_{\max}$ in terms of an unknown value α:

$$E_{\max} = \frac{1}{n - n^- - n^+} \cdot \sum_{i \in \underline{I}} \underline{x}_i + \frac{1}{n - n^- - n^+} \cdot \sum_{i \in \overline{I}} \overline{x}_i + \frac{n^+ - n^-}{n - n^- - n^+} \cdot \alpha.$$

- Then, we substitute this expression into the equation

$$\sum_{i \in \underline{I}} (\underline{x}_i - E_{\max})^{2p} + \sum_{i \in \overline{I}} (\overline{x}_i - E_{\max})^{2p} = (n - n^- - n^+) \cdot \alpha^{2p}.$$

  As a result of this substitution, we get a polynomial equation of order $\leq 2p$ in terms of the unknown value α.

- We solve this equation and find all $\leq 2p$ possible real roots α. For each of these roots, we:

  - compute the corresponding value $E_{\max}$ by using above expression for $E_{\max}$ in terms of α;
  - check whether for all $i \in I^-$, we have $\underline{x}_i \leq E_{\max} - \alpha \leq \overline{x}_i$, and whether for all $i \in I^+$, we have $\underline{x}_i \leq E_{\max} + \alpha \leq \overline{x}_i$;
  - if all these inequalities hold, we compute

$$M_{2p+1} = \frac{1}{n} \cdot \left(\sum_{i \in \underline{I}} (\underline{x}_i - E_{\max})^{2p+1} + \sum_{i \in \overline{I}} (\overline{x}_i - E_{\max})^{2p+1} + \sum_{i \in I^-} (-\alpha)^{2p+1} + \sum_{i \in I^+} \alpha^{2p+1} \right).$$

For each subdivision for which $\underline{I} = \overline{I} = \emptyset$, we:

- compute $E_{\max} - \alpha$ as $\max\limits_{i \in I^-} \underline{x}_i$, $E_{\max} + \alpha$ as $\max\limits_{i \in I^+} \underline{x}_i$, and then
- check whether $\underline{x}_i \leq E_{\max} - \alpha \leq \overline{x}_i$ for all $i \in I^-$ and whether $\underline{x}_i \leq E_{\max} + \alpha \leq \overline{x}_i$ for all $i \in I^+$;

if all these conditions are satisfied, we return $M_{2p+1} = 0$.

The largest of these returned values of M_{2p+1} is the desired largest possible value $\overline{M}_{2p+1}$ of the odd moment under interval uncertainty.

Comments. One of the steps of the above algorithm is finding all the roots of a polynomial equation of order $\leq 2p$. Two questions naturally appear: are these cases when the order is $< 2p$, and do we really need to find all the roots?

The answer to the first question is straightforward: it is possible that this equation has a lower order. For example, in the degenerate case when $\underline{I} = \overline{I} = \emptyset$, this equation takes a form $0 = 0$.

The answer to the second question is as follows. We only want the values α for which the expressions $E_{\max} - \alpha$ and $E_{\max} + \alpha$ are within the corresponding intervals. Since we know that $E_{\max}$ linearly depends on α, these restriction can be transformed into bounds on α. So, in principle, instead of finding all real roots, we can look only for the roots from the interval formed by these bounds. Other similar ideas are possible. We did not implement the above idea, and so we do not know whether it decreases the overall computation time, i.e., whether the time saved on finding extra roots is indeed larger than the additional computation time needed to find bounds for α.

5.4 Efficient Algorithm for Computing the Odd Moments under Privacy-Related Interval Uncertainty

5.4.1 Main Idea

Let us show that in the privacy case, we can have an efficient algorithm. Indeed, the above analysis has shown that if we know where the threshold values $E_{\max} - \alpha$ and $E_{\max} + \alpha$ are in relation to different intervals, and if the i-th interval $[\underline{x}_i, \overline{x}_i]$ does not contain the value $E_{\max} + \alpha$, then we know exactly where the maximum is attained.

The only case when there are several (no more than 3) options for the location of the maximizing value x_i is when the interval $[\underline{x}_i, \overline{x}_i]$ contains the value $E_{\max} + \alpha$.

Since we consider the privacy case, we only have intervals formed by the corresponding thresholds, such as 10, 20, 30, etc. years. If the value $E_{\max} + \alpha$ is different from these

thresholds, then it is bounded by two thresholds, and thus, all the intervals containing $E_{\max}+\alpha$ are equal to each other. If it so happens that the value $E_{\max}+\alpha$ exactly coincides with one of the thresholds, then we can have two threshold-bounded intervals which contain $E_{\max}+\alpha$.

5.4.2 Case when $E_{\max}+\alpha$ is Different from All the Threshold Values

Let us consider these cases one by one. If $E_{\max}+\alpha$ does not coincide with any of the threshold values, then it is different from all the endpoints $\underline{x}_i$ and $\overline{x}_i$. Thus, we have only one threshold interval $[\underline{x}_i, \overline{x}_i]$ for which $\underline{x}_i < E_{\max}+\alpha < \overline{x}_i$. For each such i, we have three possible locations of the optimizing value x_i: $\overline{x}_i$, $E_{\max}+\alpha$, and $\max(\underline{x}_i, E_{\max}-\alpha)$. There may be several ($N>1$) intervals in our database which are equal to this range. To find the optimal value of the moment, all we need to know is at how many of these intervals, we have each of the three choices. Let us denote:

- by N^+ the number of such intervals (containing $E_{\max}+\alpha$) at which $x_{i,\max} = E_{\max}+\alpha$, and
- by $\overline{N}$, the number of such intervals for which $x_{i,\max} = \overline{x}_i$.

Then, in the remaining $N - N^+ - \overline{N}$ intervals, we have $x_{i,\max} = \max(\underline{x}_i, E_{\max}-\alpha)$.

There are $\le n$ possible values of N^+; for each of these values, there is $\le n$ different possible values of $\overline{N}$. Thus, overall, we have $\le n \cdot n = n^2$ possible subdivisions of these N intervals into 3 classes. For each of these subdivisions, we can compute the corresponding values of $E_{\max}$, α, and M_{2p+1}.

5.4.3 Case when $E_{\max} + \alpha$ Coincides with One of the Threshold Values

IF $E_{\max} + \alpha$ is equal to one of the threshold values, then we have two threshold ranges that contain $E_{\max} + \alpha$: the left range $[\underline{x}_i, E_{\max} + \alpha]$ and the right range $[E_{\max} + \alpha, \overline{x}_i]$. According to our analysis of six possible cases, for each of these intervals, there are at most two possible locations of $x_{i,\max}$: $E_{\max} + \alpha$ and one other choice: $\overline{x}_i$ for the right range and $\max(E_{\max} - \alpha, \underline{x}_i)$ for the left range.

We may have several ($N_l > 1$) intervals coinciding with the left range $[\underline{x}_i, E_{\max} + \alpha]$, and we may have several ($N_r > 1$) intervals coinciding with the right range $[E_{\max} + \alpha, \overline{x}_i]$. All it matters is how many intervals we have with different choices.

Let us denote:

- by n_l^+ the number of left intervals (containing $E_{\max} + \alpha$) at which $x_{i,\max} = E_{\max} + \alpha$, and
- by n_r^+ the number of right intervals (containing $E_{\max} + \alpha$) at which $x_{i,\max} = E_{\max} + \alpha$.

Then, in the remaining $N_r - n_r^+$ right intervals, we have $x_{i,\max} = \overline{x}_i$ and in the remaining $N_l - n_l^+$ left intervals, we have $x_{i,\max} = \max(\underline{x}_i, E_{\max} - \alpha)$.

There are $\leq n$ possible values of n_l^+; for each of these values, there is $\leq n$ different possible values of n_r^+. Thus, overall, we have $\leq n \cdot n = n^2$ possible subdivisions of these N intervals into corresponding classes. For each of these subdivisions, we can compute the corresponding values of $E_{\max}$, α, and M_{2p+1}.

5.4.4 Computation Time

First, we need to describe all possible locations of the values $E_{\max} - \alpha$ and $E_{\max} + \alpha$ among the range values $t_1, \ldots, t_T$. The number of thresholds T is usually much smaller than the number of data intervals: e.g., 5 or 10. So, to estimate the computation time of the algorithm, in comparison with n, we can simply view t as a constant.

There are $2t+1$ possible locations of the value $E_{\max}-\alpha$: it can be before the first threshold t_1, exactly at this threshold, between t_1 and t_2, exactly at t_2, etc. We can describe these locations by considering a half-integer value j^- which is defined as follows:

- if $E_{\max}-\alpha=t_j$, then we take $j^-=j$;
- if $t_j<E_{\max}-\alpha<t_{j+1}$, then we take $j^-=j+\dfrac{1}{2}$;
- if $E_{\max}-\alpha<t_1$, then we take $j^-=\dfrac{1}{2}$.

For each of these locations, we can have $\le 2t+1$ possible locations of $E_{\max}+\alpha$. We can describe these locations by considering a half-integer value j^+ which is defined as follows:

- if $E_{\max}+\alpha=t_j$, then we take $j^+=j$;
- if $t_j<E_{\max}+\alpha<t_{j+1}$, then we take $j^+=j+\dfrac{1}{2}$;
- if $E_{\max}+\alpha>t_T$, then we take $j^+=T+\dfrac{1}{2}$.

Thus, overall, we have $\le (2t+1)^2=\mathcal{O}(1)$ combinations of locations of $E_{\max}-\alpha$ and $E_{\max}+\alpha$.

For each of these locations, as we have shown, we have $\le n^2$ possible locations of optimizing values $x_{i,\max}$. So, we can compute $\overline{M}_{2p+1}$ in quadratic time $\mathcal{O}(n^2)$.

5.4.5 Resulting Algorithm

In the privacy-related interval case, we have the thresholds $t_1<\ldots<t_T$, and we know that each intervals has the form $[t_j,t_{j+1}]$ for some j.

General plan. To compute $\overline{M}_{2p+1}$, we consider all possible values

$$j^-=\frac{1}{2},\ \ j^-=1,\ \ j^-=\frac{3}{2},\ \ \ldots,\ \ j^-=T.$$

For each of these values, we consider all possible values $j^+\ge j^-$ of the type

$$j^+=1,\ \ j^+=\frac{3}{2},\ \ \ldots,\ \ j^+=T,\ \ j^+=T+\frac{1}{2}.$$

First part: "deterministic" assignments (i.e., assignments for intervals in which the location of the maximizing value $x_{i,\max}$ is uniquely determined by the above analysis). For each such pair (j^-, j^+) and for each interval $[\underline{x}_i, \overline{x}_i]$ of the type $[t_{j(i)}, t_{j(i)+1}]$, we select the following value $x_{i,\max}$:

- when $j(i)+1 < j^-$ (i.e., when $\overline{x}_i < E_{\max} - \alpha$), we take $x_{i,\max} = \overline{x}_i$;
- when $j(i) < j^- \le j(i)+1 < j^+$ (second case when $\overline{x}_i < E_{\max} + \alpha$), we formally assign an expression $x_{i,\max} = E_{\max} - \alpha$, where $E_{\max}$ and α will be determined later;
- when $j^- \le j(i) < j(i)+1 < j^+$ (third case when $\overline{x}_i < E_{\max} + \alpha$), we take $x_{i,\max} = \underline{x}_i$;
- when $j(i) > j^+$ (sixth case), we take $x_{i,\max} = \overline{x}_i$.

The only remaining cases are the cases of ranges containing the threshold t_{j^+}.

Second part: "non-deterministic" assignments: case when j^+ is not an integer. If j^+ is not an integer, i.e., if $j^+ = j_0 + \dfrac{1}{2}$ for some j_0, then there is only one such range $[t_{j_0}, t_{j_0+1}]$. Let $N(j_0)$ be the total number of all the intervals $[\underline{x}_i, \overline{x}_i]$ which are equal to $[t_{j_0}, t_{j_0+1}]$. In this case, we must try all possible natural numbers N^+ and $\overline{N}$ for which $N^+\overline{N} \le N(j_0)$. Algorithmically, we have two embedded for-loops:

- we try all the values N^+ from 0 to $N(j_0)$;
- for each N^+, we try all the values $\overline{N}$ from 0 to $N(j_0) - N^+$.

For each selections of the values N^+ and $\overline{N}$, we do the following:

- to N^+ out of $N(j_0)$ indices for which $[\underline{x}_i, \overline{x}_i]$ are equal to $[t_{j_0}, t_{j_0+1}]$, we assign $x_{i,\max} = E_{\max} + \alpha$;
- to $\overline{N}$ out of $N(j_0)$ such indices, we assign $x_{i,\max} = \overline{x}_i$;
- to the remaining $N(j_0) - N^+ - \overline{N}$ indices, the assignment depends on whether we are in the fourth or in the fifth case:

- if we are in the fourth case, i.e., if $j^- > j_0$, then we assign $x_{i,\max} = E_{\max} - \alpha$;
- if we are in the fifth case, i.e., if $j^- < j_0$, then we assign $x_{i,\max} = \underline{x}_i$.

Second part: non-deterministic assignments: case when j^+ is an integer. If j^+ is an integer, then there are two ranges for which the corresponding values $x_{i,\max}$ can be different: $[t_{j^+-1}, t_{j^+}]$ and $[t^{j^+}, t_{j^++1}]$. Let N_l be the total number of all the intervals $[\underline{x}_i, \overline{x}_i]$ which are equal to $[t_{j^+-1}, t_{j^+}]$, and let N_r be the total number of all the intervals $[\underline{x}_i, \overline{x}_i]$ which are equal to $[t^{j^+}, t_{j^++1}]$. In this case, we must try all possible natural numbers n_l^+ and n_r^+ for which $n_l^+ \le N_l$ and $n_r^+ \le N_r$. Algorithmically, we have two embedded for-loops:

- we try all the values n_l^+ from 0 to N_l;
- for each n_l^+, we try all the values n_r^+ from 0 to N_r.

For each selections of the values n_l^+ and n_r^+, we do the following:

- to n_r^+ out of N_r indices for which $[\underline{x}_i, \overline{x}_i]$ are equal to $[t_{j^+}, t_{j^++1}]$, we assign $x_{i,\max} = E_{\max} + \alpha$;
- to the remaining $N_r - n_r^+$ such intervals, we assign $x_{i,\max} = \overline{x}_i$.

Also,

- to n_l^+ out of N_l indices for which $[\underline{x}_i, \overline{x}_i]$ are equal to $[t_{j^+-1}, t_{j^+}]$, we assign $x_{i,\max} = E_{\max} + \alpha$;
- to the remaining $N_l - n_l^+$ such intervals, the assignment depends on whether we are in the second or in the third case:
  - if we are in the second case, i.e., if $j^- > j_+ - 1$, then we assign $x_{i,\max} = E_{\max} - \alpha$;
  - if we are in the third case, i.e., if $j^- \le j_+ - 1$, then we assign $x_{i,\max} = \underline{x}_i$.

Third part: computing $E_{\max}$, α, and M_{2p+1}; general case. For each of these selections of the corresponding two values, we do the following:

- First, we find the coefficients of the linear expression of the unknown value $E_{\max}$ in terms of an unknown value α:

$$E_{\max} = \frac{1}{n - n^- - n^+} \cdot \sum_{i \in \underline{I}} \underline{x}_i + \frac{1}{n - n^- - n^+} \cdot \sum_{i \in \overline{I}} \overline{x}_i + \frac{n^+ - n^-}{n - n^- - n^+} \cdot \alpha,$$

  where by $\underline{I}$, we denote the set of all indices i for which we take $x_{i,\max} = \underline{x}_i$, by $\overline{I}$, we denote the set of all indices i for which we take $x_{i,\max} = \overline{x}_i$, by n^-, the number of all indices for which we selected $x_{i,\max} = E_{\max} - \alpha$, and by n^+, the number of all indices i for which we selected $x_{i,\max} = E_{\max} + \alpha$.

- Then, we substitute this expression into the equation

$$\sum_{i \in \underline{I}} (\underline{x}_i - E_{\max})^{2p} + \sum_{i \in \overline{I}} (\overline{x}_i - E_{\max})^{2p} = (n - n^- - n^+) \cdot \alpha^{2p}.$$

  As a result of this substitution, we get a polynomial equation of order $\le 2p$ in terms of the unknown value α.

- We solve this equation and find all $\le 2p$ possible roots α.

- For each of these roots, we:

  - compute the corresponding value $E_{\max}$ by using above expression for $E_{\max}$ in terms of α;
  - check whether $E_{\max} - \alpha$ is in the right place, i.e.,
    - if j^- is an integer, then $E_{\max} - \alpha = t_{j^-}$;
    - if j^- is not an integer, then $t_{\lfloor j^- \rfloor} < E_{\max} - \alpha < t_{\lceil j^- \rceil}$;
  - check whether $E_{\max} + \alpha$ is in the right place, i.e.,
    - if j^+ is an integer, then $E_{\max} + \alpha = t_{j^+}$;

$*$ if j^+ is not an integer, then $t_{\lfloor j^+ \rfloor} < E_{\max} + \alpha < t_{\lceil j^+ \rceil}$;

– if both $E_{\max} - \alpha$ and $E_{\max} + \alpha$ are in the right place, we compute

$$M_{2p+1} = \frac{1}{n} \cdot (x_{i,\max} - E_{\max})^{2p+1}.$$

Third step: computing $E_{\max}$, α, and M_{2p+1}; degenerate case. The degenerate case when all the assignments $x_{i,\max}$ are equal to either $E_{\max} - \alpha$ or to $E_{\max} + \alpha$ must be treated separately. In this case, we check whether the number of assignments of $E_{\max} - \alpha$ is equal to the number of assignments of $E_{\max} + \alpha$.

- If they are not equal, then (in line with the general algorithm) we dismiss this case as impossible.
- If they are equal, we return $M_{2p+1} = 0$.

Once one of such cases is not dismissed, we add $M_{2p+1} = 0$ to the list of possible values of M_{2p+1} and therefore, we do not need to consider any more of such degenerate cases.

Final step. The largest of thus returned values of M_{2p+1} is the desired largest possible value $\overline{M}_{2p+1}$ of the odd moment under interval uncertainty.

5.4.6 The place of the above algorithm in the general overview of statistical analysis under interval uncertainty.

The above algorithm solves an open problem from [18]. By adding the result of this algorithm, we can thus update the previously presented Table 3.2 of the computational complexity of different interval-related statistical analysis problems. In this problem, the number in the left column refers to the class as described in Table 3.1. The new result is underlined.

Here, Row 6 represents the case we are interested in: the privacy case.

#	E	V	C_{xy}	L, U, R	M_{2p}	M_{2p+1}
0	$\Theta(n)$	NP-hard	NP-hard	NP-hard	NP-hard	?
1	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
2	$\Theta(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
3	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?
4	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?
5	$\Theta(n)$	$\mathcal{O}(n^{m+1})$	?	$\mathcal{O}(n^{m+1})$	$\mathcal{O}(n^{m+1})$	?
6	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\underline{\mathcal{O}(n^2)}$
7	$\Theta(n)$	$\mathcal{O}(n \cdot \log(n))$	?	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	?

Table 5.1: Computational complexity of statistical analysis of interval data: an updated overview containing a new result about odd central moments

5.5 Auxiliary Algorithm

The above algorithm can be naturally extended to the "slightly wider intervals" case, when for some integer K, no set of K intervals has a common intersection.

Indeed, in this case, we can sort all $2n$ endpoints $\underline{x}_i$ and $\overline{x}_i$ of n given intervals into a sequence

$$x_{(0)} = -\infty \le x_{(1)} \le x_{(2)} \le \ldots \le x_{(2n)} \le x(2n+1) = +\infty.$$

For example, $x_{(1)}$ is the smallest of the lower endpoints $\underline{x}_i$, and $x_{(2n)}$ is the largest of the upper endpoints $\overline{x}_i$.

In this way, the real line is divided into $2n+1$ "zones" $(x_{(0)}, x_{(1)}]$, $[x_{(1)}, x_{(2)}]$, $\ldots$, $[x_{(2n-1)}, x_{(2n)}]$, and $[x_{(2n)}, x_{(2n+1)})$. Each of the zones $[x_{(1)}, x_{(2)}]$, $\ldots$, $[x_{(2n-1)}, x_{(2n)}]$ is fully contained in one of the intervals $[\underline{x}_i, \overline{x}_i]$. The first zone $(x_{(0)}, x_{(1)}]$ and the last zone $[x_{(2n)}, x_{(2n+1)})$ have infinite length and are not contained in any of the given n intervals.

The assignment of the values $x_{i,\max}$ depends on the zones which contain $E_{\max} - \alpha$ and $E_{\max} + \alpha$. Thus, to describe all such cases, we must select the zone that contains $E_{\max} - \alpha$

and the zone that contains $E_{\max} + \alpha$.

The value $E_{\max}$ is the arithmetic average of the values $x_{i,\max} \in [\underline{x}_i, \overline{x}_i]$; thus, it cannot exceed the largest of the upper endpoints, i.e., $E_{\max} \le x_{(2n)}$. Since $\alpha \ge 0$, we thus have $E_{\max} - \alpha \le x_{(2n)}$. So, the value $E_{\max} - \alpha$ can only be contained in one of the $2n$ intervals $(x_{(0)}, x_{(1)}]$, $[x_{(1)}, x_{(2)}]$, $\ldots$, $[x_{(2n-1)}, x_{(2n)}]$.

Similarly, the arithmetic average $E_{\max}$ of the values $x_{i,\max} \in [\underline{x}_i, \overline{x}_i]$ cannot be smaller than the smallest of the lower endpoints, i.e., $E_{\max} \ge x_{(1)}$. Since $\alpha \ge 0$, we thus have $E_{\max} + \alpha \ge x_{(1)}$. So, the value $E_{\max} + \alpha$ can only be contained in one of the $2n$ intervals $[x_{(1)}, x_{(2)}]$, $\ldots$, $[x_{(2n-1)}, x_{(2n)}]$, $[x_{(2n)}, x_{(2n+1)})$.

There are $2n$ possible zones for $E_{\max} - \alpha$ and for each such zone allocation, there are $\le 2n$ possible zones for $E_{\max} + \alpha$. So, overall, we must consider $\le (2n)^2 = \mathcal{O}(n^2)$ zone allocations.

For each zone allocation, the values $x_{i,\max}$ are uniquely determined for all the intervals $[\underline{x}_i, \overline{x}_i]$ except for the intervals which contain $E_{\max} + \alpha$. The value $E_{\max} + \alpha$ is within one of the zones.

If $E_{\max} + \alpha$ is in the last (infinite) zone $[x_{(2n)}, x_{(2n+1)})$ but not in any previous zone, then this value cannot be inside any of the intervals $[\underline{x}_i, \overline{x}_i]$ and thus, all the values $x_{i,\max}$ are uniquely determined.

If $E_{\max} + \alpha$ is in any other zone $[x_{(k)}, x_{(k+1)}]$, then, since this zone is contained in some interval $[\underline{x}_j, \overline{x}_j]$, we can only have $< K$ intervals which contain this zone – otherwise, we would have $\ge K$ intervals $[\underline{x}_i, \overline{x}_i]$ with a common intersection, in contradiction to our assumption that we have slightly wider intervals. So, for each of $\le K$ intervals, we have ≤ 3 different options of selecting $x_{i,\max}$; for all other intervals, the value $x_{i,\max}$ is determined uniquely. Thus, for this choice of zone, we must consider $< 3^K$ different cases. Since K is a constant independent on n, we have $3^K = \mathcal{O}(1)$. This means that for each of $\mathcal{O}(n^2)$ selection of zones, we must consider $\mathcal{O}(1)$ cases. In each case, we need to compute an expression for M_{2p+1} which requires $\mathcal{O}(n)$ computations. So, we get a new $\mathcal{O}(n^3)$ time algorithm for computing the endpoints of the odd moments for slightly wider interval case

as well.

5.6 Numerical Verification

In the above text, we describe an algorithm which is guaranteed to compute the exact range for the odd moments in case of privacy-related interval uncertainty, and we provide a justification for this algorithm.

To doublecheck our theoretical analysis, we programmed this algorithm and compared the result of this algorithm with the "exhaustive search". Of course, every interval contains infinitely many values, so we cannot really try all of them. Instead, we select an integer N, so that within each interval $[\underline{x}_i, \overline{x}_i]$ of width $w_i \stackrel{\text{def}}{=} \overline{x}_i - \underline{x}_i$, we test $N+1$ values

$$\underline{x}_i, \underline{x}_i + \frac{w_i}{N}, \underline{x}_i + 2 \cdot \frac{w_i}{N}, \ldots, \underline{x}_i + k_i \cdot \frac{w_i}{N}, \ldots, \underline{x}_i + N \cdot \frac{w_i}{N} = \overline{x}_i.$$

Then, we compute the odd moment M_{2p+1} for all $(N+1)^k$ possible combinations of values $x_i = \underline{x}_i + k_i \cdot \dfrac{w_i}{N} \in [\underline{x}_i, \overline{x}_i]$. Then, we take the largest of the corresponding values M_{2p+1} as an estimate for the desired upper endpoint $\overline{M}_{2p+1}$.

As $N \to \infty$, we get denser and denser coverage of the intervals and thus, better and better approximations to M_{2p+1}. In this approximation, we represent each possible value $x_i \in [\underline{x}_i, \overline{x}_i]$ by a nearest value of the type $x_i = \underline{x}_i + k_i \cdot \dfrac{w_i}{N}$. The distance from x_i to this nearest value is equal to half of the distance between the consequent discrete values, i.e., to $\leq \dfrac{w_i}{2N}$. Thus, the relative accuracy of this approximation is $\approx \dfrac{1}{2N}$. We therefore expect that the result of this discretized exhaustive search approximates the actual value of $\overline{M}_{2p+1}$ with the same relative accuracy.

This exhaustive search requires $(N+1)^n$ computation steps. Since the exponential function grows really fast, we had to limit ourselves to the cases when N and n are not too large. In our simulation, we used $N = 10$, and $n = 7$ – the case for which we need 10^7 computations of the odd moment. Each of these cases already took hours to compute, so we decided not to test more complex cases.

For $N = 10$, we expect the exhaustive search to provide us with $\approx \dfrac{1}{2N} = 5\%$ accuracy.

The above algorithm is reasonably complex; this complexity is party due to the fact that in the case of privacy-related interval uncertainty, many intervals coincide. Since in our numerical simulations, we can only test a few intervals, we decided to restrict our comparison to the case when all the input intervals are different.

We ran several numerical examples. In each example, we computed the third moment M_3 (corresponding to $p = 1$) and the fifth moment M_5 (corresponding to $p = 2$). Let us describe four representative numerical examples.

- In the first example, we take 7 consecutive intervals of the same width:

  $$[3, 4], \quad [4, 5], \quad [5, 6], \quad [6, 7], \quad [7, 8], \quad [8, 9], \quad [9, 10].$$

  For these intervals, our algorithm returns the value

  $$\overline{M}_3 = 4.507102538026902\ldots, \quad \overline{M}_5 = 85.69358667575521\ldots$$

  while exhaustive search leads to $\overline{M}_3 \approx 4.44$ and $\overline{M}_5 \approx 84.8$. Both approximate values for $\overline{M}_{2p+1}$ are indeed within 5% accuracy of our results.

- In the second example, we take 7 intervals of different width:

  $$[1, 2], \quad [2, 3], \quad [3, 6], \quad [6, 7], \quad [7, 8], \quad [8, 9], \quad [9, 11].$$

  For these intervals, our algorithm returns the value

  $$\overline{M}_3 = 10.195010806467739\ldots, \quad \overline{M}_5 = 505.4028697619265\ldots$$

  while exhaustive search leads to $\overline{M}_3 \approx 10.075$ and $\overline{M}_5 \approx 504.3$. Both approximate values for $\overline{M}_{2p+1}$ are indeed within 5% accuracy of our results.

- In the third example, we also take 7 intervals of different width:

  $$[3, 7], \quad [7, 8], \quad [8, 10], \quad [10, 11], \quad [11, 15], \quad [15, 20], \quad [20, 22].$$

For these intervals, our algorithm returns the value

$$\overline{M}_3 = 156.6407555699302\ldots, \quad \overline{M}_5 = 20576.88873911836\ldots$$

while exhaustive search leads to $\overline{M}_3 \approx 155.5$ and $\overline{M}_5 \approx 20287.8$. Both approximate values for $\overline{M}_{2p+1}$ are indeed within 5% accuracy of our results.

- Finally, in the fourth example, we take 6 intervals of different width and with gaps:

$$[2, 4], \quad [4, 5], \quad [6, 7], \quad [7, 8], \quad [8, 9], \quad [10, 12].$$

For these intervals, our algorithm returns the value

$$\overline{M}_3 = 15.594803374293013\ldots, \quad \overline{M}_5 = 587.0939291177262\ldots$$

while exhaustive search leads to $\overline{M}_3 \approx 15.5934$ and $\overline{M}_5 \approx 586.95$. Both approximate values for $\overline{M}_{2p+1}$ are indeed within 5% accuracy of our results.

In all these cases, numerical results confirm the correctness of our algorithm.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In many real-life situations, we collect a large amount of data and perform statistical analysis of this data. For example, to better understand the dependence of income, health, etc. on geographic area, gender, age, and other characteristic, researchers perform a statistical analysis of the census data. To better understand the effect of different medicine on the patients, medical researchers perform the statistical analysis of the results of medical experiments and surveys. In all these cases, it is important to preserve the privacy of the people whose data is being processed.

One way to preserve this privacy is to only store ranges (intervals) instead of the actual values: e.g., instead of the actual age, we store the range (0 to 10, 10 to 20, etc.); instead of the actual salary, we store the range of the salaries, etc. The reason why we collect all this data is that we want to perform statistical analysis, i.e., compute the values of different statistical characteristics $C(x_1, \ldots, x_n)$ such as mean, variance, higher central moments, correlation, etc. For different values x_i from the given interval ranges $[\underline{x}_i, \overline{x}_i]$, in general, we get different values of the desired characteristic $C(x_1, \ldots, x_n)$. It is desirable to find the range $\mathbf{C} = [\underline{C}, \overline{C}]$ of possible values of the desired characteristic $C(x_1, \ldots, x_n)$.

Under privacy-related interval uncertainty, efficient algorithms for computing this range were known for computing mean, variance, and, more generally, even central moments M_{2p}. For odd central moments M_{2p+1}, an efficient algorithm was previously known only for the third moment M_3. Higher odd moments, however, are also important in describing the shape of the empirical distributions. In this thesis, we design a new efficient $\mathcal{O}(n^2)$

algorithm for computing the range for odd central moments. Thus, we solve an important open problem in statistical analysis of interval data.

6.2 Future Work

Future work on this direction may include work in three directions.

First, it is desirable to extend our algorithm to other types of interval uncertainty, more general than in the privacy case.

Second, it is desirable to look for even faster (more efficient) algorithms for the privacy case; our belief that such speed-up is possible comes from the two facts: that an improvement to $\mathcal{O}(n)$ was recently attained by G. Xiang, and that, similar to the case of the third moment, it may be possible to prove that the value $E + \alpha$ cannot be attained when we look for the maximum of M_{2p+1} (the ideas on how we may be able to prove this were provided by Dr. Luis G. Valdez-Sanchez).

Third, it may be beneficial not only to provide algorithms for the case when we preserve privacy by only allowing intervals, but also to provide a trade-off between possible privacy loss and gains from the resulting statistical analysis. For this trade-off, we need to be able, e.g., to gauge the privacy loss. How to gauge privacy loss is an important but difficult problem; preliminary results in this direction are presented, e.g., in [3].

References

[1] K. Atkinson, *Elementary Numerical Analysis*, John Wiley & Sons Inc, 1993.

[2] M. Bishop, *Computer Security: Art and Science*, Addison Wesley, 2002.

[3] V. Chirayat, L. Longpré, V. Kreinovich, "Measuring Privacy Loss in Statistical Databases", In: H. Leung and G. Pighizzini (Eds.), Proceedings of the Workshop on Descriptional Complexity of Formal Systems DCFS 2006, Las Cruces, New Mexico, June 21–23, 2006, pp. 16–25.

[4] T. H. Cormen et al., *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2001.

[5] F. A. Cowell, "Grouping bounds for inequality measures under alternative informational assumptions", *J. of Econometrics*, 1991, Vol. 48, pp. 1–14.

[6] E. Dantsin, V. Kreinovich, A. Wolpert, and G. Xiang, "Population Variance under Interval Uncertainty: A New Algorithm", *Reliable Computing*, 2006, Vol. 12, No. 4, pp. 273–280.

[7] T. Dalenius, "Finding a needle in a haystack – or identifying anonymous census record", *Journal of Official Statistics*, 1986, Vol. 2, No. 3, pp. 329–336.

[8] D. E. R. D. Denning, *Cryptography and data security*, Addison-Wesley, Reading, MA, 1982.

[9] G. Duncan and D. Lambert, "The risk of disclosure for microdata", *Proc. of the Bureau of the Census Third Annual Research Conference*, Bureau of the Census, Washington, DC, 1987, pp. 263–274.

[10] G. Duncan and S. Mukherjee, "Microdata disclosure limitation in statistical databases: query size and random sample query control", *Prof. 1991 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, May 20–22, 1991.

[11] I. Fellegi, "On the question of statistical confidentiality", *Journal of the American Statistical Association*, 1972, pp. 7–18.

[12] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles, "Computing Variance for Interval Data is NP-Hard", *ACM SIGACT News*, 2002, Vol. 33, No. 2, pp. 108–118.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman and Company, 1979.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, Springer-Verlag, London, 2001.

[15] V. Kreinovich and L. Longpré, Computational complexity and feasibility of data processing and interval computations, with extension to cases when we have partial information about probabilities, In: V. Brattka, M. Schroeder, K. Weihrauch, and N. Zhong, editors, *Proc. Conf. on Computability and Complexity in Analysis CCA'2003*, Cincinnati, Ohio, USA, August 28–30, 2003, pp. 19–54.

[16] V. Kreinovich, L. Longpré, S. Ferson, and L. Ginzburg, *Computing Higher Central Moments for Interval Data*, University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-03-14b, 2004, http://www.cs.utep.edu/vladik/2003/tr03-14b.pdf

[17] V. Kreinovich, L. Longpré, S. A. Starks, G. Xiang, J. Beck, R. Kandathi, A. Nayak, S. Ferson, and J. Hajagos, "Interval Versions of Statistical Techniques, with Applications to Environmental Analysis, Bioinformatics, and Privacy in Statistical Databases",

Journal of Computational and Applied Mathematics, 2007, Vol. 199, No. 2, pp. 418–423.

[18] V. Kreinovich, G. Xiang, S. A. Starks, L. Longpre, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos, "Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity", *Reliable Computing*, 2006, Vol. 12, No. 6, pp. 471–501.

[19] J. Kim, "A method for limiting disclosure of microdata based on random noise and transformation", *Proceedings of the Section on Survey Research Methods of the American Statistical Association*, 1986, pp. 370–374.

[20] N. Kirkendall et al., *Report on Statistical Disclosure Limitations Methodology*, Office of Management and Budget, Washington, DC, Statistical Policy Working Paper No. 22, 1994.

[21] A. T. Langewisch and F. F. Choobineh, "Man and variance bounds and propagation for ill-specified random variables", *IEEE Trans. SMC*, 2004, Vol. 34, No. 4, pp. 494–506.

[22] L. Longpré, V. Kreinovich, E. Freudenthal, M. Ceberio, F. Modave, N. Baijal, W. Chen, V. Chirayath, G. Xiang, and J. I. Vargas, "Privacy: Protecting, Processing, and Measuring Loss", *Abstracts of the 2005 South Central Information Security Symposium SCISS'05*, Austin, Texas, April 30, 2005, p. 2.

[23] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed*, McGraw Hill, 2005.

[24] F. Modave, M. Ceberio, X. Wang, G. Xiang, O. Garay, R. Ramirez, R. Tejeda, "Comparison of Computer Attacks: An Application of Interval-based Fuzzy Integration", Proceedings of the 24th International Conference of the North American Fuzzy In-

formation Processing Society NAFIPS'2005, Ann Arbor, Michigan, June 22-25, 2005, pp. 676–681.

[25] M. Morgenstern, "Security and inference in multilevel database and knowledge base systems", *Proc. of the ACM SIGMOD Conference*, 1987, pp. 357–373.

[26] H. T. Nguyen, V. Kreinovich, V. I. Gorodetski, V. M. Nesterov, and A. L. Touloupiev, "Applications of Interval-Valued Degrees of Belief: A Survey", In: A. Touloupiev (ed.), *Information Technologies and Intellectual Methods*, Vol. 3 (IT&IM'3), St. Petersburg Institute for Information and Automation of Russian Academy of Sciences (SPIIRAS), 1999, pp. 6–61 (in Russian).

[27] Office of Technology Assessment, *Protecting privacy in computerized medical information*, US Government Printing Office, Washington, DC, 1993.

[28] M. Palley and J. Siminoff, "Regression methodology based disclosure of a statistical database", *Proceedings of the Section on Survey Research Methods of the American Statistical Association*, 1986, pp. 382–387.

[29] C. P. Pfleeger, S. L. Pfleeger *Security in Computing*, Prentice Hall, 2003.

[30] S. Rabinovich, *Measurement Errors and Uncertainties: Theory and Practice*, Springer-Verlag, New York, 2005.

[31] S. Ross, *A First Course in Probability*, Prentice Hall, 2002.

[32] M. Sipser, *Introduction to the Theory of Computation*, Thomson, Boston, Massachusetts, 2006.

[33] T. Su and G. Ozsoyoglu, "Controlling FD and MVD inference in multilevel relational database systems", *IEEE Transactions on Knowledge and Data Engineering*, 1991, Vol. 3, pp. 474–485.

[34] L. Sweeney, "Weaving technology and policy together to maintain confidentiality", *Journal of Law, Medicine and Ethics*, 1997, Vol. 25, pp. 98–110.

[35] L. Sweeney, "Datafly: a system for providing anonymity in medical data", In: T. Y. Lin and S. Qian (eds.), *Database Security XI: Status and Prospects*, Elsevier, Amsterdam, 1998.

[36] S. A. Vavasis, *Nonlinear Optimization: Complexity Issues*, Oxford University Press, New York, 1991.

[37] L. Willenborg and T. De Waal, *Statistical disclosure control in practice*, Springer Verlag, New York, 1996.

[38] G Xiang, S. Starks, V. Kreinovich,, L. Longpré, " New Algorithms for Statistical Analysis of Interval Data," Proceedings of the Workshop on State-of-the-Art in Scientific Computing PARA'04, Lyngby, Denmark, June 20–23, 2004, Vol. 1, pp. 123–129.

Appendix A

Source Code

This appendix provides the source code for our implementation and the instructions for using this source code. This source code was written and tested for Java version 1.5.0_02. We tested it on versions 1.4.1 and 1.4.2, it works on these versions as well. (We did not test it on older versions of Java, because these versions are not used anymore.)

The algorithm is contained in the class `OddMoments`. The method that runs the algorithm is `runAlgorithm`, its signature is:

```
public double runAlgorithm (ArrayList list, int moment)
```

To run this method, you need to create all the input intervals as objects of type `Interval`; a constructor for the class `Interval` takes as inputs the endpoints of the interval and returns the interval object with these endpoints. For example, if we want to create an interval called `first` with the value $[1.0, 2.0]$, and an interval called `second` with the value $[2.0, 3.0]$, we must call the corresponding constructor as follows:

```
Interval first = new Interval(1.0,2.0);
Interval second = new Interval(2.0,3.0);
```

To make sure that our program can handle lists of arbitrary size, we describe all the intervals not as an array, but as an `ArrayList`. To handle ArrayLists, we must import the corresponding library

```
import java.util.Arraylist;
```

Then, we can create an empty `ArrayList`, and use the standard `add` operation to add intervals. For example, to create an `ArrayList` named `list` which contains the intervals `first` and `second`, we need to write the following code

```
ArrayList list = new ArrayList();
list.add(first);
list.add(second);
```

As usual in Java, we can simultaneously create intervals and add them:

```
ArrayList list = new ArrayList();
list.add(new Interval(1.0,2.0));
list.add(new Interval(2.0,3.0));
```

The value `moment` is the odd number $2p + 1$. In the attached code, we can only pass the values 3 or 5.

In this program, we use Newton's method to find the roots α of the corresponding polynomial. For that, we describe a method `f(...)` which, given α, computes the value of the corresponding polynomial, and a method `fPrime(...)` which computes the derivative of this polynomial (with respect to α). To compute odd moments of higher orders, we need to correspondingly adjust the methods `f` and `fPrime`.

The method `runAlgorithm` returns the largest possible value $\overline{M}_{2p+1}$ of the corresponding moment M_{2p+1} over the given intervals.

```
import java.util.Collections;
import java.util.ArrayList;
public class OddMoments {

//-------------------------------------------------------------
//
// Instance and Class Variables Declaration
//
//-------------------------------------------------------------

   /** A representative value for E - alpha */
   public static final double OMEGA = -1.856397;
   /** A representative value for E + alpha */
   public static final double OMEGA_PRIME = -2.0036954;
   /** The value passed to Newton's method to begin approximating when
       f( x ) = 0 */
   public static final double INITIAL_GUESS = 0.25;
   /** The tolerance for Newton's method. The iteration will stop when
       f ( x ) < TOLERANCE */
   public static final double TOLERANCE = 0.0001;
   /** Indicates whether to print values at run time to debug or not */
   private boolean debug;

//-------------------------------------------------------------
//
// Constructors
//
//-------------------------------------------------------------
```

```
    /**
     * Creates a new instance of this class
     *
     * @param boolean debug : Indicates whether to print values to debug
     *                        at run time
     *
     * @return A new instance of this class
     */
    public OddMoments(boolean debug) {
        this.debug = debug;
    }

    /**
     * Creates a new instance of this class. If this constructor is
     * invoked, debug mode is automatically turned off
     *
     * @param none
     *
     * @return A new instance of this class
     */
    public OddMoments( ) {
        this.debug = false;
    }

//-------------------------------------------------------------
//
// Main Method
```

```
//
//-------------------------------------------------------

   /**
    * Runs the application
    *
    * @param String[ ] args : The arguments to run the application
    *
    * @return void
    */
   public static void main(String[ ] args) {
      OddMoments om = new OddMoments(false);
      ArrayList intervals = om.getIntervals( );
      System.out.println
  ("\n****************** Calculating Third Moment ******************\n");
      System.out.println
  ("Maximum Computed by Algorithm: " + om.runAlgorithm(intervals, 3));
      System.out.println
      ("Maximum Computed by Brute-Force: " +
        om.computeByBruteForceMaximum(intervals.toArray( ), 3));

      System.out.println
  ("\n****************** Calculating Fifth Moment ******************\n");
      System.out.println
  ("Maximum Computed by Algorithm: " + om.runAlgorithm(intervals, 5));
      System.out.println
      ("Maximum Computed by Brute-Force: " +
        om.computeByBruteForceMaximum(intervals.toArray( ), 5));
```

```
   } // End of main


//-------------------------------------------------------------
//
// getIntervals
//
//-------------------------------------------------------------

   /**
    * Retrieves the set of intervals to compute the sample odd moment
    *
    * @param none
    *
    * @return The intervals to be evaluated
    */
   public static ArrayList getIntervals( ) {
      ArrayList intervals =  new ArrayList( );

      // Add intervals in the following fashion ...
      intervals.add(new Interval(1, 2));
      // Add as many as desired ....
      // ........

      return intervals;

   } // End of getIntervals
```

```
// ------------------------------------------------------
//
//  Algorithm
//
// ------------------------------------------------------


 /**
  * Run the algorithm to compute the maximum value for the range. For
  * this version, we are testing the skewness or third moment with the
  * general algorithm. Moreover, we are assuming the privacy case.
  * Hence, once the intervals are sorted - if the intervals are
  * continous - this same list are the zones. For instance, for the
  * following intervals:
  * [1, 3] [2, 4] [3, 5]
  * the zones would be
  * [1, 2] [2, 3] [3, 4] [4, 5]
  * However, for the privacy case, two intervals either coincide or
  * differ. Hence, we are assuming the case above is not possible to
  * occur
  *
  * @param ArrayList list : The intervals used to calculate the maximum
  *                         sample odd moment
  * @param int moment : The moment to calculate the maximum
  *
  * @return The maximum value attained of the corresponding odd moment
  */
```

```
public double runAlgorithm(ArrayList list, int moment) {

   // This variable holds the largest value computed for the range
   double maximum = 0.0;

   // Run the actual algorithm
   Collections.sort(list); // Sort intervals in lexicographic order

   // The set of zones of intervals to iterate to compute the
   // maximum value
   Object[ ] interval = list.toArray( );

   Object[ ] zone = getZones(interval);

   int length = zone.length;            // The number of zones

   if (debug) {
       System.out.println("Zones sorted: ");
       for (int i = 0; i < length; i++)
           System.out.println(zone[i] + "\n\n");
   }

  boolean firstPass = true;
  for (int i = 0; i < length; i++) {
     firstPass = true;
     for(int j = i + 1; j < length; j++) {
        int jCopy = j;              // For debugging purposes
```

```
// The unknown values n and n', which corresponds to the
// number of indices that contain E - alpha and E + alpha,
// respectively
short nPrime = 0;
short nDoublePrime = 0;

for(int k = 0; k < interval.length; k++) {

   if (((Interval)interval[k]).getUpperBound( ) <=
      ((Interval)zone[i]).getLowerBound( ))
      ((Interval)interval[k]).setMaximum(
        ((Interval)interval[k]).getUpperBound( ));

   if (((Interval)interval[k]).getLowerBound( ) <=
      ((Interval)zone[i]).getLowerBound( ) &&
      ((Interval)interval[k]).getUpperBound( ) >=
      ((Interval)zone[i]).getUpperBound( )) {
      ((Interval)interval[k]).setMaximum(OMEGA);
       nPrime ++;
   }

   if (((Interval)zone[i]).getUpperBound( ) <=
      ((Interval)interval[k]).getLowerBound( ) &&
      ((Interval)interval[k]).getUpperBound( ) <=
      ((Interval)zone[j]).getLowerBound( ))
      ((Interval)interval[k]).setMaximum(
        ((Interval)interval[k]).getLowerBound( ));
```

```
    if (interval[k].equals(zone[j]) && !firstPass)
       ((Interval)interval[k]).setMaximum(
         ((Interval)interval[k]).getUpperBound( ));

    if (((Interval)interval[k]).getLowerBound( ) >=
       ((Interval)zone[j]).getUpperBound( ))
       ((Interval)interval[k]).setMaximum(
         ((Interval)interval[k]).getUpperBound( ));



    if (interval[k].equals(zone[j]) && firstPass) {
       ((Interval)interval[k]).setMaximum(OMEGA_PRIME);
       nDoublePrime ++;

       if ( (j + 1) == length) {
          // Set j = i so that when the loop is over j is
          // incremented, giving j = i + 1 as desired
          j = i ;
          firstPass = false;
       }
    }
    if (debug)
System.out.println("Maximum of interval[ " + k + " ] : " +
                   ((Interval)interval[k]).getMaximum( ));
 }
 // Computes the odd moment and checks whether this value is
 // greater than  the previous sample odd moment computed
 double temp = computeSampleOddMoment(interval, zone, nPrime,
```

```
                                                    nDoublePrime, i, jCopy,
                                                    moment);
            if (temp > maximum || (maximum == 0.0 &&
                !Double.isNaN(temp)))
               maximum = temp;
            if (debug)
               System.out.println("\nMaximum Computed: " + temp + '\n');
         }
      }
      return maximum;

   } // End of runAlgorithm

//-----------------------------------------------------------
//
// Private Methods
//
//-----------------------------------------------------------

   /**
    * Computes the sample odd moments using the information provided
    *
    * @param Object[ ] intervals : The intervals used to compute the
    *                              sample odd moment
    * @param short n : The number of intervals that were assigned the
    *                  value E - alpha
    * @param short nPrime : The number of intervals that were assigned
    *                       the value E + alpha
```

```
* @param int i : The index that points to the interval that possibly
*                contains the value E - alpha
* @param int j : The index that points to the interval that possibly
*                contains the value E + alpha
* @param int moment : The moment to be computed. This number is
*                     assumed to be odd
*
* @return The sample odd moment computed if the information provided
*         is correct, i.e. the value E - alpha and E + alpha are
*         within the intervals that contains the values OMEGA and
*         OMEGA_PRIME. If this condition is not satisfied, -1 is
*         returned
*/
private double computeSampleOddMoment(Object[ ] intervals,
                                      Object[ ] zones,
                                      short nPrime,
                                      short nDoublePrime, int i,
                                      int j, int moment) {
   int n = intervals.length;
   double sum = computeSumOfMaximums(intervals, 0, 1);

   double divisor = (double)(n - nPrime - nDoublePrime);

   if (debug)
       System.out.println("Sum: " + sum);

   // Obtain the coefficients for E and alpha
   double[ ] coefficients = getCoefficients(sum / divisor,
```

```
                        (double)(nDoublePrime - nPrime) / divisor,
                        intervals, nPrime, nDoublePrime, moment);
double E = coefficients[0];
double alpha = coefficients[1];

// Debugging
if (debug) {
    System.out.println
    ("E : " + E  + "\t alpha: " + alpha);
    System.out.println
    ("E - alpha: " + (E - alpha) + "\tE + alpha: " + (E + alpha));
    System.out.println("zone[i] : " + zones[i]);
    System.out.println("zone[j] : " + zones[j]);
}

// Checks whether E - alpha and E + alpha are indeed in the
// indicated zones
if (((Interval)zones[i]).getLowerBound( ) <= (E - alpha) &&
    ((Interval)zones[i]).getUpperBound( ) >= (E - alpha))
   ((Interval)zones[i]).setMaximum(E - alpha);
else return Double.NaN;

if (((Interval)zones[j]).getMaximum( ) == OMEGA_PRIME) {
     if (((Interval)zones[j]).getLowerBound( ) <= (E + alpha) &&
         ((Interval)zones[j]).getUpperBound( ) >= (E + alpha))
         ((Interval)zones[j]).setMaximum(E + alpha);
     else return Double.NaN;
}
```

```
// At this point, we know the values E - alpha and E + alpha were
// indeed in the specified zones. Now we compute the actual sample
// odd moment with the information given

int index = 0;
for(int k = 0; k < n; k++) {
   if(zones[k].equals(intervals[index]))
      ((Interval)intervals[index++]).setMaximum
                              (((Interval)zones[k]).getMaximum( ));
}

n = intervals.length;
double e = 0.0;
for(int x = 0; x < n; x++) {
   e += ((Interval)intervals[x]).getMaximum( );
}

if (debug) {
    System.out.println("Value computed for E: " + E);
    System.out.println("Real Value: " + e);
}

e /= n;

double oddMoment = 0.0;         // Stores the sample odd moment

for(int k = 0; k < n; k++)
```

```
        oddMoment += Math.pow(
                     ((Interval)intervals[k]).getMaximum( ) - E,
                      moment);

    if (debug) {
       for(int k = 0; k < n; k++)
          System.out.println("interval[ " + k + " ] : " +
                             ((Interval)intervals[k]).getMaximum( ));
    }

    return oddMoment / n;

 } // End of computeSampleOddMoment

/**
 * Retrieves the zones to run the algorithm
 *
 * @param Object[ ] intervals : The intervals used to get the zones
 *
 * @return The zones to run the algorithm
 */
 private Object[ ] getZones(Object[ ] intervals) {
    ArrayList zones = new ArrayList( );
    for(int i = 0; i < intervals.length; i++) {
       zones.add(intervals[i]);
       if((i + 1) < intervals.length &&
            ((Interval)intervals[i]).getUpperBound( ) <
            ((Interval)intervals[i + 1]).getLowerBound( ))
```

```
                zones.add(new Interval
                      (((Interval)intervals[i]).getUpperBound( ),
                      ((Interval)intervals[i + 1]).getLowerBound( )));
    } // for

    return zones.toArray( );

} // End of getZones

/**
* Computes the coefficients for E and alpha. To compute such values,
* we consider the formula of E in the following terms:
*      E = (k / n) + (n' / n) * (E - alpha) + (n'' / n) * (E + alpha)
*      n * E = k + n' * E - n' * alpha + n'' * E + n'' * alpha
*      n * E - n' * E - n'' * E = k + -n' * alpha + n'' * alpha
*      (n - n' - n'') * E = k + (n'' - n') * alpha
*      E = (k + (n'' - n') * alpha) / (n - n' - n'')
*
* where
*       k = sum of maximums that were known,
*       n' = number of intervals assigned the value OMEGA
*       n'' = number of intervals assigned the value OMEGA_PRIME
*
* @param double sum : The value of the sum of the maximum of the
*                     intervals which were known. This parameter
*                     corresponds to the constant on the above
*                     equation
* @param double alphaConstant : The value of the constant
```

```
*                                multiplying alpha
* @param Object[ ] intervals : The intervals to compute the
*                              neccessary computations in order to
*                              find the coefficients for E and alpha
* @param int nPrime : The number of indices that are assigned the
*                     value E - alpha
* @param int nDoublePrime : The number of indices that are assigned
*                           the value E + alpha
* @param int moment : The sample odd moment which is being calculated
*
* @return The coefficients for E and alpha
*/
private double[ ] getCoefficients(double sum, double alphaConstant,
                                  Object[ ] intervals, int nPrime,
                                  int nDoublePrime, int moment) {

  // Variables to store final results
  double[ ] result  = new double[2];
  double E;

  int n = intervals.length;

  if (debug)
      System.out.println
      ("Sum: " + sum + "\talphaConstant: " + alphaConstant);

  switch(moment) {
```

```
            case 3:
               // Compute the values of a, b, and c to solve the quadratic
               // equation
               double a = (n - nPrime - nDoublePrime) *
                          (1 - Math.pow(alphaConstant, 2));
               double b = 2 * computeSumOfMaximums
                              (intervals, sum * -1, 1) * alphaConstant;
               double c = -1 * computeSumOfMaximums(intervals, sum * -1, 2);

               // Compute the coefficients for alpha
               double[ ] alphas = solveQuadraticEquation(a, b, c);
               result[1] = alphas[0];
               break;

            case 5:
               double alpha = newtonMethod(intervals, sum, nPrime,
                                           nDoublePrime, alphaConstant);
               result[1] = alpha;
               break;
         }

         E = sum + alphaConstant * result[1];
         result[0] = E;
         return result;

      } // End of getCoefficients

//-------------------------------------------------------------
```

```
// Third Moment
//-------------------------------------------------------------

   /**
    * Solve a quadratic equation given constants a, b, c
    *
    * @param double a : The constant a in the formula of the quadratic
    *                   equation
    * @param double b : The constant b in the formula of the quadratic
    *                   equation
    * @param double c : The constant c in the formula of the quadratic
    *                   equation
    *
    * @return An array containing the solution of the quadratic equation
    */
   private double[ ] solveQuadraticEquation(double a, double b,
                                            double c) {
      // Debugging
       if (debug)
          System.out.println("a: " + a + " \tb: " + b + " \tc: " + c);
      double firstSolution = (-b + Math.sqrt(Math.pow(b, 2) -
                                             4 *a * c)) / (2 * a);
      double secondSolution = (-b - Math.sqrt(Math.pow(b, 2) -
                                              4 *a * c)) / (2 * a);
      double[ ] ans = {firstSolution, secondSolution};
      return ans;

   } // End of solveQuadraticEquation
```

```
//-------------------------------------------------------------
// Fifth Moment
//-------------------------------------------------------------

   /**
    * Evaluates a function at a given point. To obtain this expression,
    * note that E = sum + k * alpha, where sum is the sum of the maximum
    * values which are known and k is a constant. We use this expression
    * into alpha ^ 4 = 1 / n (\sum_{i = 1}^n (x_i - E) ^ 4 )
    * The variable of this equation is alpha, the rest of the parameters
    * are constants
    *
    * @param Object[ ] intervals : The intervals used to compute the
    *                              value of the equation
    * @param double sum : The constant which is add on the expression of E
    * @param int nPrime : The number of intervals which are assigned the
    *                     value E - alpha
    * @param int nDoublePrime : The number of intervals which are
    *                           assigned the value E + alpha
    * @param double k : The constant that multiples alpha
    * @param double alpha : The variable of the expression
    *
    * @return The value of the expression evaluated at alpha, i.e.
    *         f( alpha )
    */
   private static double f(Object[ ] intervals, double sum, int nPrime,
                           int nDoublePrime, double k, double alpha) {
```

```
    double n = (double)intervals.length;
    double fourth = computeSumOfMaximums(intervals, 0, 4);
    double third = computeSumOfMaximums(intervals, 0, 3);
    double square = computeSumOfMaximums(intervals, 0, 2);
    double E = sum + k * alpha;

    // These are terms of the formula of E in terms of alpha
    double a = fourth + nPrime * Math.pow(E - alpha, 4) +
               nDoublePrime * Math.pow(E + alpha, 4);
    double b = third + nPrime * Math.pow(E - alpha, 3) +
               nDoublePrime * Math.pow(E + alpha, 3);
    double c = square + nPrime * Math.pow(E - alpha, 2) +
               nDoublePrime * Math.pow(E + alpha, 2);
    double d = 3 * Math.pow(E, 4);
    double e = Math.pow(alpha, 4);

    return (1 / n) * (a - 4 * E * b + 6 * Math.pow(E, 2) * c) - d - e;

} // End of f


/**
 * Evaluates a function at a given point. The function we are
 * evaluating is the derivative of the function f, defined above
 *
 * @param Object[ ] intervals : The intervals used to compute the
 *                              value of the equation
 * @param double sum : The constant which is add on the expression of E
```

```
 * @param int nPrime : The number of intervals which are assigned the
 *                     value E - alpha
 * @param int nDoublePrime : The number of intervals which are
 *                           assigned the value E + alpha
 * @param double k : The constant that multiplies alpha
 * @param double alpha : The variable of the expression
 *
 * @return The value of the expression evaluated at alpha, i.e.
 *         fPrime( alpha )
 *
 * @see f
 */
private static double fPrime(Object[ ] intervals, double sum,
                             int nPrime, int nDoublePrime, double k,
                             double alpha) {
   double n = (double)intervals.length;
   double third = computeSumOfMaximums(intervals, 0, 3);
   double square = computeSumOfMaximums(intervals, 0, 2);
   double E = sum + k * alpha;
   // Terms used to compute f'
   double a = 4 * nPrime * Math.pow(E - alpha, 3) * (k - 1) +
              4 * nDoublePrime * Math.pow(E + alpha, 3) * (k + 1);
   double b = third + nPrime * Math.pow(E - alpha, 3) +
              nDoublePrime * Math.pow(E + alpha, 3);
   double c = 3 * nPrime * Math.pow(E - alpha, 2) * (k - 1) +
              3 * nDoublePrime * Math.pow(E + alpha, 2) * (k + 1);
   double d = square + nPrime * Math.pow(E - alpha, 2) +
              nDoublePrime * Math.pow(E + alpha, 2);
```

```
  double e = 2 * nPrime * (E - alpha) * (k - 1) +
             2 * nDoublePrime * (E + alpha) * (k + 1);
  double f = 12 * Math.pow(E, 3) * k;
  double g = 4 * Math.pow(E, 3);

  return (1 / n) * (a - ( 4  * k * b + 4 * E * c) + ( 12 * E * k * d
                   + 6 * Math.pow(E, 2) * e)) - f - g;

} // End of fPrime

/**
 * Computes the coefficient of alpha such that f ( alpha ) = 0. This
 * is approximated using Newton's method
 *
 * @param Object[ ] intervals : The intervals used to compute the
 *                              value of the equation
 * @param double sum : The constant which is add on the expression of E
 * @param int nPrime : The number of intervals which are assigned the
 *                     value E - alpha
 * @param int nDoublePrime : The number of intervals which are
 *                           assigned the value E + alpha
 * @param double k : The constant that multiples alpha
 *
 * @return The coefficient of alpha that makes f ( alpha ) = 0
 *
 * @see f
 * @see fPrime
 */
```

```
   private static double newtonMethod(Object[ ] intervals, double sum,
                                      int nPrime, int nDoublePrime,
                                      double k) {
      double alpha = INITIAL_GUESS;

      while (Math.abs(
              f(intervals, sum, nPrime, nDoublePrime, k, alpha)) >
             TOLERANCE) {
         alpha = alpha - (f(intervals, sum, nPrime, nDoublePrime, k,alpha)
                          /
                          fPrime(intervals, sum, nPrime, nDoublePrime, k,
                                 alpha));
      }
      return alpha;

   } // End of newtonMethod


//-------------------------------------------------------------
// Auxilary Methods to compute moments
//-------------------------------------------------------------

   /**
    * Computes the sum of the maximum of the intervals which are knwon.
    * Such sum is raised to a given exponent, i.e. computes
    * \sum_{ i \in N } (x_i - c)^m, where m is the exponent, c is a
    * constant, and N is the set of values which are known
    *
```

```
 * @param Object[ ] intervals - The intervals used to compute the sum
 *                              of the maximum values
 * @param double c : The constant added to the x_i's, i.e.
 *                   sum(x_i + c) ^ K
 * @param int power - The power to be raised each of the known values
 *                    x_i
 *
 * @return The sum of the maximum values which are kwnon raised to
 *         power given
 */
 private static double computeSumOfMaximums(Object[ ] intervals,
                                            double c, int power) {
   double result = 0;
   for (int i = 0; i < intervals.length; i++) {
      if (((Interval)intervals[i]).getMaximum( ) != OMEGA &&
          ((Interval)intervals[i]).getMaximum( ) != OMEGA_PRIME)
         result += Math.pow(((Interval)intervals[i]).getMaximum( ) + c,
                            power);
   }
   return result;

} // End of computeSumOfMaximums


/**
 * Computes the sample odd moment by brute force. In other words, the
 * maximum is computed checking several possible combinations. For an
 * interval, the maximum is checked at lower bound, the upper bound,
```

```
 * and 8 more points in between. So it requires 10 ^ n possible
 * combinations. The purpose of this method is to check if the result
 * given by the algorithm is indeed the maximum value
 *
 * @param Object[ ] intervals : The intervals used to compute the
 *                              maximum sample odd moment
 * @param int moment : The sample odd moment to be computed
 *
 * @return The maximum sample odd moment computed checking all
 *         possible combinations
 */
private double computeByBruteForceMaximum(Object[ ] intervals,
                                          int moment) {
    int n = intervals.length;
    int tenToN = (int)Math.pow(10, n);
    double max = 0.0;
    String maxDecimal = "";

    double[ ][ ] maxIntervals = new double[n][10];
    for(int i = 0; i < n; i++) {
        double step = (((Interval)intervals[i]).getUpperBound( ) -
                      ((Interval)intervals[i]).getLowerBound( )) / 9;
        double value = ((Interval)intervals[i]).getLowerBound( );
        for(int j = 0; j < 10; j++) {
            maxIntervals[i][j] = value;
            value += step;
        }
    }
```

```
for(int i = 0; i < tenToN; i++) {
    String decimal = i + "";
    int padds = n - decimal.length( );

    for(int j = 0; j < padds; j++)
        decimal = '0' + decimal;

    double E = 0.0;
    double temp = 0.0;

    for(int j = 0; j < decimal.length( ); j++) {
        E += maxIntervals[j][Integer.parseInt(
                            decimal.charAt(j) + "")];
    }

    E /= n;

    for(int j = 0; j < decimal.length( ); j++)
        temp += Math.pow(maxIntervals[j][Integer.parseInt
                        (decimal.charAt(j) + "")] - E, moment);

    temp /= n;

    if (temp > max || i == 0) {
      max = temp;
      maxDecimal = decimal;
    }
```

```
       } // End of loop

       return max;

   } // End of computeByBruteForceMaximum

} // End of OddMoments


//-------------------------------------------------------------
//
// Class Interval
//
//-------------------------------------------------------------


public class Interval implements Comparable {

//-------------------------------------------------------------
//
// Instance and Class Variables Definitions
//
//-------------------------------------------------------------

    /** The upper bound of the interval */
    private double upperBound;
    /** The lower bound of the interval */
    private double lowerBound;
```

```
    /** The maximum value attained in this interval */
    private double maximum;

//-------------------------------------------------------------
//
// Constructor
//
//-------------------------------------------------------------

   /**
    * Creates a new instance of this class
    *
    * @param double upperBound : The upper bound of this interval
    * @param double lowerBound : The lower bound of this interval
    *
    * @return A new instance of this class
    */
   public Interval(double upperBound, double lowerBound) {
      this.upperBound = Math.max(upperBound, lowerBound);
      this.lowerBound = Math.min(lowerBound, upperBound);
      this.maximum = 0;
   }

//-------------------------------------------------------------
//
// Public Methods
//
//-------------------------------------------------------------
```

```
/**
 * Retrieves the upper bound of this interval
 *
 * @param none
 *
 * @return The upper bound of the interval
 */
public double getUpperBound( ) {
   return upperBound;
}

/**
 * Retrieves the lower bound of this interval
 *
 * @param none
 *
 * @return The lower bound of the interval
 */
public double getLowerBound( ) {
   return lowerBound;
}

/**
 * Retrieves the maximum value attained in this interval
 *
 * @param none
 *
```

```
 * @return The maximum value of the interval
 */
public double getMaximum( ) {
   return maximum;
}

/**
 * Set the maximum value attained in this interval. This value is
 * the maximum value that can be attained at this interval.
 *
 * @param double maximum : The maximum value for the sample odd
 *                         moments
 *
 * @return void
 */
public void setMaximum(double maximum) {
   this.maximum = maximum;
}

/**
 * Retrieves a String representation of this Object
 *
 * @param none
 *
 * @return A string displaying the bounds of the interval
 */
public String toString( ) {
   return "Lower Bound: " + lowerBound + " Upper Bound: " + upperBound;
```

```
 }

 /**
  * Compares two intervals. The intervals are compared in order to
  * be sorted in lexicographic order. Hence, intervalA > intervalB iff
  * intervalA.lowerBound >= intervalB.lowerBound &&
  * intervalA.upperBound >= intervalB.upperBound. For this equation to
  * hold, we assume no interval is a proper subset of another interval
  *
  * @param Object obj : The object to be compared with this interval
  *
  * @return A positive integer is this interval is greater than obj.
  *         A negative integer if this interval is less than obj. Zero
  *         if both intervals have the same bounds
  */
 public int compareTo(Object obj) {
    if (!(obj instanceof Interval))
       return -1; // non-comparable object ....
    Interval temp = (Interval)obj;
    if (this.lowerBound == temp.lowerBound &&
        temp.upperBound == this.upperBound)
return 0;
    if (this.lowerBound == temp.lowerBound) {
       if (temp.upperBound > this.upperBound)
          return -2;
    }
    if (temp.lowerBound > this.lowerBound)
       return -2;
```

```
      else return 2;
   } // End of compareTo

    public boolean equals(Object obj) {
if(!(obj instanceof Interval))
    return false;

Interval o = (Interval)obj;

return this.lowerBound == o.lowerBound &&
               this.upperBound == o.upperBound;
    } // End of equals

} // End of Interval
```

Curriculum Vitae

Ivan Vargas was born on October 2, 1981. He is the second son of Miguel Vargas and Concepcion Morales. He entered The University of Texas at El Paso (UTEP) in the Fall of 2000. While pursuing his Bachelor's degree in Computer Science, he worked as a Teaching Assistant, and as a Math Tutor for the Tutoring and Learning Center. He received his Bachelor's degree in Computer Science in the Spring of 2005.

In the Fall of 2005, he entered the Graduate School of The University of Texas at El Paso. Supported by the National Science Foundation Bridge-to-the-Doctorate Fellowship program, he pursued his Master's degree in Computer Science. He is a member of the UTEP chapter of Upsilon Pi Epsilon Computer Science Honor Society.

Permanent address: 9951 Rosa M Richardson
El Paso, Texas 79927