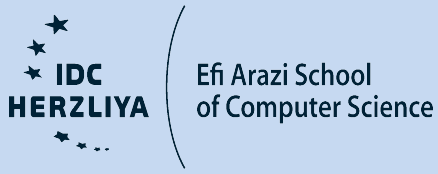


The Interdisciplinary Center, Herzlia
Efi Arazi School of Computer Science
M.Sc. program - Research Track

# Logrank Test: Statistical Stability & MPC Protocol

by
**Anat Samohi**

M.Sc. dissertation, submitted in partial fulfillment of the requirements for the M.Sc. degree, research track, School of Computer Science
Reichman University (IDC Herzliya)

June 2022

This work was carried out under the supervision of Prof. Zohar Yakhini from the Efi Arazi School of Computer Science, Reichman University, and Dr. Adi Akavia from the Department of Computer Science, University of Haifa.

# Abstract

This research revolves around the Logrank Test algorithm, a survival analysis and evaluation method. This thesis has two parts.

In the first part, we present a secure multi-party protocol for the Logrank test and prove its correctness. The protocol can be implemented by known cryptographic tools, chosen according to a required level of security and an adversary model.
Our protocol computes a value that is close to the Logrank test result under some reasonable assumptions. This approximation is empirically demonstrated.
We discuss the problem of false inputs in MPC protocols and how these frauds can be avoided. We develop and analyze an additional version of the Logrank MPC protocol that provides partial protection against false inputs.

In the second part, we analyze the uncertainty in the Logrank result that may arise from labeling errors. We present a novel algorithm for efficiently calculating a stability interval around the original Logrank p-value with the corresponding correctness proof.
A stability interval contains all possible results that would be obtained under labeling errors with defined constraints.
A paper describing this research was published in the journal Bioinformatics in 2021, and is attached to this thesis.

# Contents

# Chapter 1

# Introduction

This work revolves around the Logrank Test algorithm. Logrank is one of several survival analysis tools, used to measure and compare performance and failures over time. The comparison is between several categories, typically two. An illustrative example is the comparison between two competing treatments for a severe medical condition.

Logrank Test is useful for less dramatic comparisons as well: reliability of mechanical devices, life span of electric bulbs, or the effectiveness of different website designs in keeping the users engaged. Still, the most common use of Logrank Test for research purposes is in the medical field.

## 1.1 Motivation And Overview

When conducting an experiment using Logrank Test to analyse data, it is best, from the statistical point of view, to have the largest possible dataset. Therefore, if several parties conduct the same experiment independently they can obtain better results by sharing their data and performing the Logrank analysis on the merged dataset. By merging the data, a significant p-value can be reached with relatively moderate (less radical) experimental

results. In practice, however, research institutes may be reluctant to share any data, which is considered to be a very valuable resource. Privacy considerations therefore can lead to compromised accuracy.

In the first part of this thesis we present a **secure multiparty computation (MPC) protocol for Logrank Test**. Such a solution can solve the conflict, not compromising on either results accuracy nor on data privacy requirements. Such a solution can facilitate cooperation between institutes and organizations and thus lead to higher quality results without risking exposure of valuable data.

The Logrank test is based on several assumptions that support the validity of the calculations. It is naturally assumed, implicitly, that no errors occur in the labeling of the samples. That is - that the mapping between samples and groups is perfectly correct. In practice, however, test results are affected when considering some errors in the original labeling. To estimate this effect, we defined the **Logrank Stability Interval** which bounds the uncertainty that arises from labeling errors in Logrank test.

In the second part of this thesis we present a methodology to compute stability intervals for Logrank. A paper describing this work was published in July, 2021:
https://doi.org/10.1093/bioinformatics/btab693

## 1.2 Logrank Test - Background

### 1.2.1 Events

The most important concept in Survival Analysis is the **event**. There are two types of events:

1. Failure - the "sad point of no return". It can be either a death of a patient, a break down of electric device, or a burnout of a light bulb. A failure can also be a positive event, such as achieving the goal of a lower-cholesterol diet.

2. Censorship - the point in time in which the instance track is lost. Censorship may be due to connection loss, or due to a fixed termination date of the study.

An instance is considered **at risk** on time $t$ if at this time neither failure nor censorship have been yet observed for this instance.

### 1.2.2 The Survival Function

Consider a set of iid (independent and identically distributed) instances with a time-of-failure attribute denoted by T.
$P(T = t)$ is the distribution of T.
The *survival function* is defined as:

$$S(t) = 1 - P(T < t) = 1 - F_T(t)$$

$F_T(t)$ is the cumulative distribution function of T.
Example: If t=100 years and events represent death, then S(t=100) denotes the probability of living beyond 100 years. (see [6], Chapter 1.3)

### 1.2.3 The Logrank test

*Logrank Test* [14],[6]Chapter 2.6, is a hypothesis test procedure that compares the survival distributions of two groups of samples. These two groups differ by the feature we would like to inspect. For example, if the comparison is between two medical treatments, all patients treated with treatment A are associated to one group, and all patients treated with treatment B are associated to the other group.
Logrank test evaluates the p-value, the probability of the observations under some null model. Usually this null model assumes that survival times are derived from independent and identically distributed sampling.

The data is organized in two groups of instances, group A and group B. In each group the instances are iid and randomly selected (in order to avoid undesired bias). The assumption we would like to test and then possibly reject is:

$\forall t : S_A(t) \equiv S_B(t)$ , when $S_A()$ and $S_B()$ are the respective survival functions in each group.

In fact, the test evaluates the probability of an empirical observation, based on the assumption that $S_A \equiv S_B(t)$. This assumption is called *Null Assumption* and denoted as $H_0$.

The Logrank Test presumes the probability for an event to occur in any given time interval is equal for the two groups (directly follows from $H_0$).

The following semantics and notation for the Logrank Test will be used throughout the remainder of this thesis:

- Consider a time-and-event dataset $D$ (survival data) such that each subject (entry) has a partition labeling (groups A or B).
- Let $j = 1, ..., J$ be the distinct times of observed failure events in either group.
- Let $n_{A,j}$, $n_{B,j}$ be the number of subjects *at risk* (who have not yet failed nor have been censored) at the time of occurrence of the $j$'th failure in each group, respectively.
- $n_j = n_{A,j} + n_{B,j}$ is the total number of subjects at risk at time $j$.
- Let $O_{A,j}$, $O_{B,j}$ be the random variables representing the observed number of failures in each group at time $j$.
- Let $o_{A,j}$ and $o_{B,j}$ denote the number of failures observed at time $j$ in groups A and B, respectively. Let $o_j = o_{A,j} + o_{B,j}$ denote the number of failures observed at time $j$ in both groups.
- Let T be the time of failure of a subject. $P(T = t)$ is the probability distribution function of T. The survival function is defined as $S(t) = 1 - P(T < t) = 1 - F(t)$, where $F(t)$ is the cumulative distribution function.

- As mentioned above, the null model in Logrank testing assumes that the survival functions of the two groups are identical, $S_A(t) \equiv S_B(t)$. The null model also assumes that the variables $O_{A,j}$ (for all time intervals) are collectively independent.

- Under the null model the following applies: $O_{A,j} \sim HG(n_j, n_{A,j}, o_j)$, where HG stands for Hyper-Geometric distribution [8] [1].

  Similarly for group $B$ we have $O_{B,j} \sim HG(n_j, n_{B,j}, o_j)$.

- The expected value and the variance of $O_{A,j}$ are given by:

$$E_{A,j} = \frac{n_{A,j}}{n_j} o_j$$

$$Variace = V_{A,j} = \frac{n_{A,j}}{n_j} o_j \Big(\frac{n_j - o_j}{n_j}\Big)\Big(\frac{n_j - n_{A,j}}{n_j - 1}\Big)$$

  Similarly for group B.

- The Logrank statistic is defined as:

$$Z_A = \frac{O - E}{\sqrt{V}}$$

  Where:

$$O = \sum_{j=1}^{J} o_{A,j} \quad E = \sum_{j=1}^{J} E_{A,j} \quad V = \sum_{j=1}^{J} V_{A,j}$$

If $J$ is sufficiently large and the partition into $A$ and $B$ is reasonably balanced then $Z_A$ is approximately distributed as $N(0, 1)$. The p-value evaluated for the dataset $D$ uses the observed value of $Z_A$ and this stated normal approximation.

## 1.3 Secure Multiparty Computation - Background

### 1.3.1 MPC Protocols

Generally, every secure multiparty computation (MPC)[12][22][18] protocol is designed to address the following situation: Consider a set of parties such that each party has an input it desires to keep private. The parties use a predefined protocol to jointly compute a function of their inputs. As an example, consider 10 individuals, each having an input $x_i$. All participants wish to evaluate $\Sigma_{i=1}^{10} x_i$ without exposing their own private inputs.

In MPC protocols a certain set of **security properties** must be preserved[11]:

- Privacy - no information about the inputs may be revealed except for the output and quantities that directly follow.
- Independence of inputs - no party can choose it's input according to other inputs
- Correctness - the protocol computes the function correctly
- Fairness - if one party is exposed to it's output as defined by the protocol, all parties must be exposed to their outputs we well.

While the privacy, correctness and independence of inputs are essential, the fairness requirement can be relaxed in some cases. This kind of compromise supports a more efficient protocols in terms of performance.

### 1.3.2 The Adversary Model

Every secure MPC protocol is designed for a certain adversarial model. The adversary is considered to be an entity that has full control over one or more parties in the setting. As such, the adversary has the full knowledge and perspective ("point of view") of the corrupted parties. The adversary sees their inputs, their coin tossing results (when such are involved), and the messages received during the protocol run.

Two different models of adversarial **behaviour** can be considered:

- Semi-honest - the adversary follows the protocol (i.e. "plays by the rules"), while only attacking the privacy and attempting to extract information about the inputs of the other parties.

- Malicious - the adversary does not follow the protocol, and attempts to attack privacy as well as correctness, independence of inputs and fairness.

The **computation power** of the adversary is also significant when designing a protocol. The adversary can have either polynomial computation power or unbounded computational power.

Another characteristic of the design is related to the timing of corruption. In a **static setting**, the parties are corrupted before the protocol is initiated. In an **adaptive setting**, the adversary can corrupt parties during the protocol's run, based on past events.

These three assumptions - the adversary's behaviour, the adversary's computational power and it's ability to manipulate the setting - are the cornerstone of defining and evaluating any secure MPC protocol.

### 1.3.3 Security in MPC Protocols

A secure MPC protocol is a protocol that preserves all security properties - privacy, correctness, fairness and independence of inputs - with respect to the adversary model. Nevertheless, in this work we present a modular protocol, which can be adequate for different adversary models.

Every function $f()$ can be evaluated by a secure protocol as was first demonstrated by Yao in 1986 [22]. Yao's construction showed how any function based on a boolean circuit and calculated on the inputs of two parties can be securely computed.
BenOr, Goldwasser and Wigderson published in 1988 a protocol for securely evaluating any function, under the assumption that the majority of the parties are not corrupted.

In theory, these constructions can securely calculate any function. However, using them as part of valid solutions to a concrete problem usually results in poor performance and high complexity. In the real world, it is much more efficient to design dedicated protocols for specific tasks.

### Notation

Let $\bar{X} = (x_1, x_2, ..., x_N)$ and $\bar{Y} = (y_1, y_2, ..., y_N)$ be random vectors.
The notation $\bar{X} \equiv^C \bar{Y}$ means that the distribution of vector $\bar{X}$ and the distribution of vector $\bar{Y}$ are **close**.
**Close** can mean:

- Identical distributions. Insures perfect secrecy.
- *Statistical Indistinguishability.* The distributions are not identical, but for every sample value $\bar{C_0}$ : $Pr[\bar{X} = \bar{C_0}] \approx Pr[\bar{Y} = \bar{C_0}]$ . Statistical closeness insures statistical secrecy.
- *Computational Indistinguishability.* For every PPT (probabilistic polynomial time) algorithm A, the probability that A distinguishes between the distributions of the two vectors $\bar{X}, \bar{Y}$ is negligible. Computational Indistinguishability insures computational secrecy.

### Definition

There are formal comprehensive definitions for a secure MPC protocol which are adequate for different types of adversaries. But for the examples that will be presented in the following chapters, a simplified informal definition will suffice. The simple case involves two parties in a static setting and a semi-honest adversary.
So instead of presenting the formal definition, consider the following informal description[11].
**A secure two party protocol against a semi-honest adversary:**
Let $f(x, y) = \Big(f_1(x, y), f_2(x, y)\Big)$ be a functionality that can be computed by a $PPT$ algo-

rithm. $\pi$ is a two-party protocol of $f(x, y)$. $x$ is the input of party $P_1$ and $y$ is the input of party $P_2$. Similarly, $f_1(x, y)$ is the result exposed to party $P_1$ and $f_2(x, y)$ is the result exposed to party $P_2$. The security parameter is $n$.

$\pi$ securely computes $f$ in the presence of a static semi-honest adversary if there exist $PPT$ algorithms $S_1$ and $S_2$ (Simulators) such that for every set of values $(x, y, n)$ the joint distributions of the vectors resulting from running the protocol and the simulators are **close**:

$$\left\{\left(S_1\Big(1^n, x, f_1(x, y)\Big), f(x, y)\right)\right\} \equiv^C \left\{\left(view_1^\pi(x, y, n), f(x, y)\right)\right\}$$

and

$$\left\{\left(S_2\Big(1^n, y, f_2(x, y)\Big), f(x, y)\right)\right\} \equiv^C \left\{\left(view_2^\pi(x, y, n), f(x, y)\right)\right\}$$

where:

$view_i^\pi(x, y, n)$ = (input of party i, coin tosses of party i, messages received by party i)

Note that the above requirement also forces the following condition on the marginal distributions:

$$\left\{S_1\Big(1^n, x, f_1(x, y)\Big)\right\} \equiv^C \left\{view_1^\pi(x, y, n)\right\}$$

and

$$\left\{S_2\Big(1^n, y, f_2(x, y)\Big)\right\} \equiv^C \left\{view_2^\pi(x, y, n)\right\}$$

The algorithms $S_1$ and $S_1$ can be thought of as **Simulators**. A simulator is an algorithm that resembles a fair protocol run. Intuitively, a protocol is secure if a simulator with no input could perfectly imitate any party during the protocol such that the other party will never notice a difference.

# Part I

# CoPPSA - Collaborative Privacy-Preserving Survival Analysis

# Chapter 2

# Previous Work

Survival analysis tools and Logrank Test in particular are used in clinical studies for diverse measurements, such as the efficacy of medical treatments or for comparing life expectancy between populations (see 1.2.2 and 1.2.3). Due to the important role of survival analysis in medical research, several secure protocols for survival analysis were published in recent years. In this section we review three state of the art published reports [17] [**logrank˙prot**] [13].

Privacy Preserving is a general concept of protecting the privacy of the subjects analyzed in a dataset. In 2017 Nguyen & Hui [17] suggested an efficient **Differential Privacy** solution for survival analysis, under the assumption that the data follows the Weibull distribution.

Differential Privacy is a rigorous mathematical definition of privacy-preservation.

Consider a computation taken over some dataset. If a computation suffices differential privacy then adding an individual's information to the dataset (or removing one out of it) cannot conspicuously change the distribution of the computation result. Differential privacy does not achieve the level of security that a secure MPC protocol guarantees. While in MPC protocol no information regarding the input (beyond what is known form the output) can

leak, differential privacy only guarantees that the output does not reveal information about any entry in the dataset.

In 2011 Tingting Chen and Sheng Zhong published an MPC protocol for Logrank Test which is based on the Secure Sum operation[5]. However, the Secure Sum is not secure against a setting with more than one corrupted party - as will be shown below. Since the Logrank MPC protocol is based on this operation, the protocol is also not secure for settings with more than one corrupted party.

**Observation**: Secure sum is not secure in a setting with more than one corrupted party.

*Proof.* Consider a set of $K$ parties $P_1, ..., P_K$. Each party $i$ has an input $x_i$ ,$i = 0, 1, ...K$.

Let $C$ be a known parameter such that $x_i < C$ for $i = 0, 1, ...K$.

Let $f(x_1, x_2, ..., x_K) = x_1 + x_2 + ... + x_K$

The Secure Sum operation is as follows:

1. $P_1$ samples $R_1 \sim Uni[0, CK - 1]$ and sends $m_1 = (x_1 + R_1) \mod (CK)$ to $P_2$
2. for $j = 2, ..., K - 1$: $P_j$ Sends $m_j = (x_j + m_{j-1}) \mod (CK)$ to $P_{j+1}$.
3. $P_K$ Sends $m_K = (x_K + m_{K-1}) \mod (CK)$ to $P_1$.
4. $P_1$ sends $output = (m_K - R_1) \mod (CK)$ to all other parties.

For $t > 1$ this protocol is not secure.

For example, if the adversary corrupted parties $P_2$ and $P_4$, then it can calculate $x_3$ with full confidence:

$$P\Big[X_3 = (m_3 - m_2) \mod (CK) | m_3, m_2\Big] = 1$$

Which proves the observation. □

Furthermore, even though Chen-Zhong's protocol is aimed to calculate Z-score, it reveals additional information which should remain hidden.
Chen-Zhong's protocol has two versions. In the first version both the total numbers of instances "at risk" and the total number of failures for each time interval are exposed. In the second version of the protocol the total amount of failures in the target group is exposed. Although this can be considered as a reasonable compromise in some cases, it is not true for all scenarios.

On January 2021 von Maltitz et al described a high-performance implementation for a secure MPC protocol for Logrank test[13]. Their implementation is based on FRESCO - Framework for Efficient Secure Computation - a Java-based repository for writing applications based on secure computation.

Von Maltitz et al demonstrated that with a common hardware and networking framework, a secure MPC protocol for Logrank can run within a reasonable time . Specifically, their protocol took 20 minutes for 300 items merged dataset divided to 3 subsets. In their paper they described the dataset merging process:

1. Each party defines a list of failure times.
2. All parties use MPC protocol to create one merged sorted list of all failure times. By the end of this step the only intermediate information that was made available to all parties was the merged list of failure times.
3. Each party creates a table with the Logrank data: the amount of subject "at risk", the total amount of failures, and the amount of failures in the target group. Each entry in the table is associated with a time in the merged list.
4. Each party encrypts it's table, as generated above.
5. The parties use an additive MPC protocol to marge all ciphertext to one encrypted table.

The encrypted merged table is an encrypted Logrank timeline. The consequent operations in the calculation are now executed straight forward on the encrypted data. Note that the encryption therefore needs to support division and roots. In fact, for each time interval (for each encrypted entry in the merged table) the expected value and variance are calculated over encrypted data. That requires a number of division operations that is linear to the number of failures.

Von Maltitz et al's solution is accurate and fully secure, although it exposes the time intervals of the merged tables to all parties. Their measurements show that the most influential factors on time performance were the number of "heavy" arithmetic operations (such as divisions), and network latency.

# Chapter 3

# The Protocol

The naive way to perform the Logrank calculation in MPC would be to first encrypt the data table of each party, then to merge the encrypted data, and finally to perform all of the Lograk process on encrypted dataset. Unfortunately this method is extremely inefficient. Logrank calculation requires both sorting and division operations. The number of required division operation is linear in the number of time intervals in the data; this has proven to be a highly resource-demanding process, to a point of infeasibility when dealing with even mildly large amounts of encrypted data.

We could provide a new MPC protocol designed for a specific adversary model, but we would like to propose a more general solution for the efficiency problem. We offer an alternative calculation for Logrank which is based on changing the Logrank process so that the parties do most of the required operations before the data is encrypted. This way we minimize the number of operations on the ciphertext.

Our solution actually computes a variant of the standard approximation. In section 4 we discuss statistical aspects related to the difference between Logrank and CoPPSA

In this chapter we present the CoPPSA protocol and discuss it's correctness and it's security. CoPPSA can be implemented based on a variety of known MPC tools, like BGW, Homomorphic Encryption, or others.

## 3.1 CoPPSA - An MPC Protocol For Logrank Test

The notations follow the notations of section 1.2.3. $o_{A,j}(i), E_{A,j}(i), V_{A,j}(i)$ represent $o_{A,j}, E_{A,j}, V_{A,j}$ as calculated by party $i$.

Let $Z$ be the result of a standard Logrank Test applied on a dataset $D$. CoPPSA calculates a different quantity. Let $Z^*$ denote the CoPPSA protocol result on the same dataset $D$.

$$Z^* = \frac{\sum_{i=1}^{N} O(i) - E(i)}{\sqrt{\sum_{i=1}^{N} V(i)}}$$

Logrank calculations are performed over the field of real numbers. Therefore, the accuracy of calculation is governed by the accuracy of the inputs. The parameter $b$ represents the number of decimal digits after the decimal point. For calculating Logrank results with a given accuracy the parties must agree on the parameter $b$ and calculate $(O - E, V)$ accordingly.

Since Logrank calculates Z-score, a reasonable value can be $b = 2$ or $b = 3$.

1. For each party i, $i = 1, 2...., N$:

   - The party calculates the vector $\Big(o_{A,j}(i), E_{A,j}(i), V_{A,j}(i)\Big)$

     for each time interval j.

     The time intervals $1 \leq j \leq J_i$ are associated with the party's own dataset.

     $J_i$ denotes the number of time intervals in the dataset of the party.

   - The party calculates over the field of the real numbers $\mathbb{R}$:

   $$O(i) = \sum_{j=1}^{J_i} o_{A,j}(i) \quad E(i) = \sum_{j=1}^{J_i} E_{A,j}(i) \quad V(i) = \sum_{j=1}^{J_i} V_{A,j}(i)$$

   - The party calculates $O(i) - E(i)$ and $V(i)$ as rounded integers:

   $$m_1(i) = \lfloor 10^b \cdot (O(i) - E(i)) \rfloor, \quad m_2(i) = \lfloor 10^b \cdot V(i) \rfloor$$

     The party then forms the message:

   $$M(i) = (m_1(i), m_2(i))$$

2. The Parties use MPC to jointly calculate a value over a cyclic field $\mathbb{Z}_p$

   (see Section 3.1.2):

   $$D = (\sum_{i=1}^{N} m_1(i)) \, mod \, p$$

   $$U = (\sum_{i=1}^{N} m_2(i)) \, mod \, p$$

   (D,U) is the common output of all parties.

3. Each party calculates over $\mathbb{R}$:

   $$Z^* = \frac{D \cdot 10^{-b}}{\sqrt{U \cdot 10^{-b}}}$$

**Algorithm 1:** CoPPSA protocol

Note that all the secure MPC operations are in Phase 2, on calculating (D, U). All other calculations are done locally by each party, without further communication.

In terms of MPC, phase 2 requires a secure calculation of sums. This can be implemented by any MPC protocol that supports summing over $\mathbb{Z}_p$ and is suitable for the required adversary model.

Note that no sorting operations and no division operations are required.

### 3.1.1 Correctness

Now we show that $Z^*$ calculates the required value:

In phase 3 we get:

$$Z^* = \frac{D \cdot 10^{-b}}{\sqrt{U \cdot 10^{-b}}} = \frac{(\sum_{i=1}^{N} m_1(i)) \; mod \; p \cdot 10^{-b}}{\sqrt{(\sum_{i=1}^{N} m_2(i)) \; mod \; p \cdot 10^{-b}}} = \frac{\left(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor\right) \mod p \cdot 10^{-b}}{\sqrt{(\sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor) \mod p \cdot 10^{-b}}}$$

$$= \frac{\left(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor\right) \cdot 10^{-b}}{\sqrt{(\sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor) \cdot 10^{-b}}} \approx \frac{\sum_{i=1}^{N} O(i) - E(i)}{\sqrt{\sum_{i=1}^{N} V(i)}}$$

Where we use the construction of $p$ (see section 3.1.2) and where the approximation sign $\approx$ is due to rounding.

### 3.1.2 The Field $\mathbb{Z}_p$

Now we show how to select the field size $p$ .

Note that phases 1 and 3 imply the following requirements:

1. $m_1(i), m_2(i) \in \mathbb{Z}_p$
2. $D = \left(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor\right) \mod p = \left(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor\right)$
3. $U = (\sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor) \mod p = \sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor$

Let $o_{max}$ be the maximum number of failures possible in a party's database. We will show that if $p > N \cdot o_{max} \cdot 10^b$ then the requirements are fulfilled.

Consider a case where N parties wish to choose a cyclic field $\mathbb{Z}_p$ for the protocol.

The input, to the protocol, of each party is $\Big(O(i)-E(i), V(i)\Big)=\Big(\sum_{j=1}^{J_i}(o_{A,j}(i)-E_{A,j}(i)), \sum_{j=1}^{J_i} V_{A,j}(i)\Big)$

The required accuracy is $10^{-b}$ ($b$ decimal digits).

The common output: $(D,U) = \Big(\sum_{i=1}^{N}\lfloor 10^b \cdot (O(i)-E(i))\rfloor, \sum_{i=1}^{N}\lfloor 10^b \cdot V(i)\rfloor\Big)$

$$\Big(\sum_{i=1}^{N}\lfloor 10^b \cdot (O(i)-E(i))\rfloor, \sum_{i=1}^{N}\lfloor 10^b \cdot V(i)\rfloor\Big) \leq \Big(10^b \sum_{i=1}^{N}(O(i)-E(i)), 10^b(\sum_{i=1}^{N} V(i))\Big)$$

$$D = (\sum_{i=1}^{N} m_1(i)) \; mod \, p \leq 10^b \sum_{i=1}^{N} O(i) - E(i)$$

$$U = (\sum_{i=1}^{N} m_2(i)) \; mod \, p \;\; \leq 10^b (\sum_{i=1}^{N} V(i))$$

1. $o(i)$ - the number of failures of party i. The parties should agree over some value $o_{max}$ such that $\forall i = 1,..,N : o(i) \leq o_{max}$.

2. $$Var(o_{A,j}) = \frac{n_{A,j} n_{B,j}}{n_j^2} \frac{o_j(n_j - o_j)}{(n_j - 1)} \leq \frac{1}{4} \frac{o_j(n_j - o_j)}{(n_j - 1)} \leq \frac{o_j(n_j - 1)}{4(n_j - 1)} \leq \frac{o_j}{4}$$

   Therefore:

   $$V(i) = \sum_{j=1}^{J_i} Var(o_{A,j}) \leq \frac{1}{4} \sum o_j(i) = \frac{1}{4} o(i)$$

   Conclusion: $O(i) - E(i) \leq o(i) \leq o_{max}, V(i) \leq \frac{1}{4} o(i) \leq \frac{1}{4} o_{max}$

3. $$(O(i) - E(i)) \cdot 10^b \leq o_{max} \cdot 10^b \quad , \quad V(i) \cdot 10^b \leq \frac{1}{4} o_{max} \cdot 10^b$$

   $$U = (\sum_{i=1}^{N} m_2(i)) \; mod \, p = \sum_{i=1}^{N} V(i) \cdot 10^b \leq \frac{N}{4} \cdot 10^b \cdot o_{max}$$

   $$D = (\sum_{i=1}^{N} m_1(i)) \; mod \, p = \sum_{i=1}^{N} (O(i) - E(i)) \cdot 10^b \leq N \cdot o_{max} \cdot 10^b$$

So, we need to set the field $\mathbb{Z}_p$ such that $N \cdot o_{max} \cdot 10^b < p$

### 3.1.3 CoPPSA Security

As mentioned above, all the secure MPC operations are in phase 2.

The functionality that is being securely calculates is:

$$D = (\sum_{i=1}^{N} m_1(i)) \, mod \, p$$

$$U = (\sum_{i=1}^{N} m_2(i)) \, mod \, p$$

$(m_1(i), m_2(i))$ is the private input of party $i$ and $(D, U)$ is the common output of all parties. The security of CoPPSA is therefore directly inherited from the security of the chosen MPC protocol for phase 2. Since the MPC tool is not specified in the protocol definition, the level of security is flexible, as well as the adversary model. The MPC tool should be chosen according to the requirements. In that sense the level of security depends on the implementation.

For example, consider the following implementation of phase 2 with Homomorphic Encryption.

Consider an additively Homomorphic Encryption scheme $\pi = (Gen, Enc, Dec, \oplus)$ such that:

$$\forall m, m' \in \mathbb{Z}_p \quad Dec_{sk}(Enc_{pk}(m) \oplus Enc_{pk}(m')) = m + m' \mod p$$

Let $\mathbb{Z}_p$ be the plaintext field. We assume that $Enc_{pk}()$ is a function from $\mathbb{Z}_p$ to some larger field $\mathbb{Z}_q$.

Consider a case in which $N$ hospitals conducted the same experiment, and they wish to use CoPPSA to securely calculate $Z^*$. They use an implementation of CoPPSA in which there is an evaluation server and a key-holder server. This implementation of the protocol relies on the assumption that the adversary cannot corrupt both servers at the same time.

**Steps:**

1. The key-holder server calculates $Gen(1^k) \rightarrow (pk, sk)$ and publishes $pk$ to all hospitals.

2. Each hospital $i$ encrypts the message $M(i)$ and sends the ciphertext to the evaluation server.

$$Enc_{pk}(M(i)) \rightarrow C(i), \quad C(i) = \Big(c_1(i) \mod q, \quad c_2(i) \mod q\Big)$$

   Note that $c_1(i)$ and $c_2(i)$ are both in $\mathbb{Z}_q$

3. The evaluation server performs the evaluation and sends the result to the key-holder server.

$$\tilde{D} = \oplus_{i=1}^{N} c_1(i) \quad \tilde{U} = \oplus_{i=1}^{N} c_3(i)$$

4. The key-holder decrypts D and U and sends the results to all hospitals.

   From the construction of $\mathbb{Z}_p$:

$$Dec_{sk}(\tilde{D}) = Dec_{sk}\Big(\oplus_{i=1}^{N} c_1(i)\Big) = \Big(\sum_{i=1}^{N} m_1(i)\Big) \mod p = D \mod p = D$$

$$Dec_{sk}(\tilde{U}) = Dec_{sk}\Big(\oplus_{i=1}^{N} c_3(i)\Big) = \Big(\sum_{i=1}^{N} m_2(i)\Big) \mod p = U \mod p = U$$

It is easy to see that under the assumption regarding the servers, the security is kept.

**Discussion:**

It is important to note that the common output $(D, U)$ reveals information other than $Z*$. The output $(D, U)$ is defined as follows:

$$D = (\sum_{i=1}^{N} m_1(i)) \, mod \, p$$

$$U = (\sum_{i=1}^{N} m_2(i)) \, mod \, p$$

Using CoPPSA force the participants to share $(D, U)$ instead of sharing $Z$. In some cases, revealing $(D, U)$ instead of $Z$ may be considered as a "deal breaker".
For example, if there are only two parties participating, the output is

$$D = (m_1(1) + m_1(2)) \, mod \, p$$

$$U = (m_2(1) + m_2(2)) \, mod \, p$$

In this case, there is no advantage in using MPC tool for phase 2 at all. The inputs $m_1(1), m_1(2), m_2(1), m_2(2)$ that should have remained private can be directly calculated by each party from the output. In Von Maltitz et al solution, or any other solution that define $Z$ as the common outputs, there is no such a problem, because the functionality that is being securely calculated is $Z$ and not $(D, U)$.

### 3.1.4 Simulation

A simple example of phases 2 and 3 of the protocol is available at
https://github.com/asamohi/Logrank_submodule.git
In this repository you can find a single-threaded simulation. The implementation is based on Homomorphic Encryption (CKKS scheme). This example contains clients (parties) and two servers. One server provides keys creation and decryption, and the other server provides evaluation. We assume that the adversary does not corrupt both servers at the same time.

#### Space consumption

In our simulation each client sends encrypted data with size 750KB. The evaluation server sends encrypted data to the key-holder server, which has the same size: 750KB.
We ran the simulation with several numbers of clients with the following running times:

| Number of clients | Running time, in seconds |
|---|---|
| 3 | 1 |
| 10 | 3.25 |
| 100 | 31 |
| 1000 | 307 |

As expected, the time is linearly proportional to the number of parties. Note that our protocol is completely independent of the amount of instances.

# Chapter 4

# Statistical Analysis And Methods

In this chapter we will discuss several statistical aspects of CoPPSA. We will present the difference between CoPPSA and standard Logrank Test by experimenting with an actual data and also by a thought experiment. We will justify the correctness of CoPPSA as a Survival Analysis comparison tool. We will empirically demonstrate that the results of CoPPSA converge to those of Logrank test.

## 4.1 Is $Z^*$ Equivalent To $Z$?

As mentioned in the previous chapters, CoPPSA evaluates $Z^*$ – which is a different function than the standard Logrank result $Z$. Figure 4.1 will assist in better understanding the source of the differences between the two protocols. The top histogram in the figure shows a simulation of a dataset of size 200 (corresponding to 200 patients). The dataset is the MAINZ cohort [19]. We divided the data according to the sub-type - Luminal A vs not Luminal A - and define the target group to be Luminal A. A standard Logrank calculation on this dataset results with a p-value of 0.0142. This result is marked with a red line

The 200 subjects of the dataset were then uniformly randomly split into 5 subsets. Then

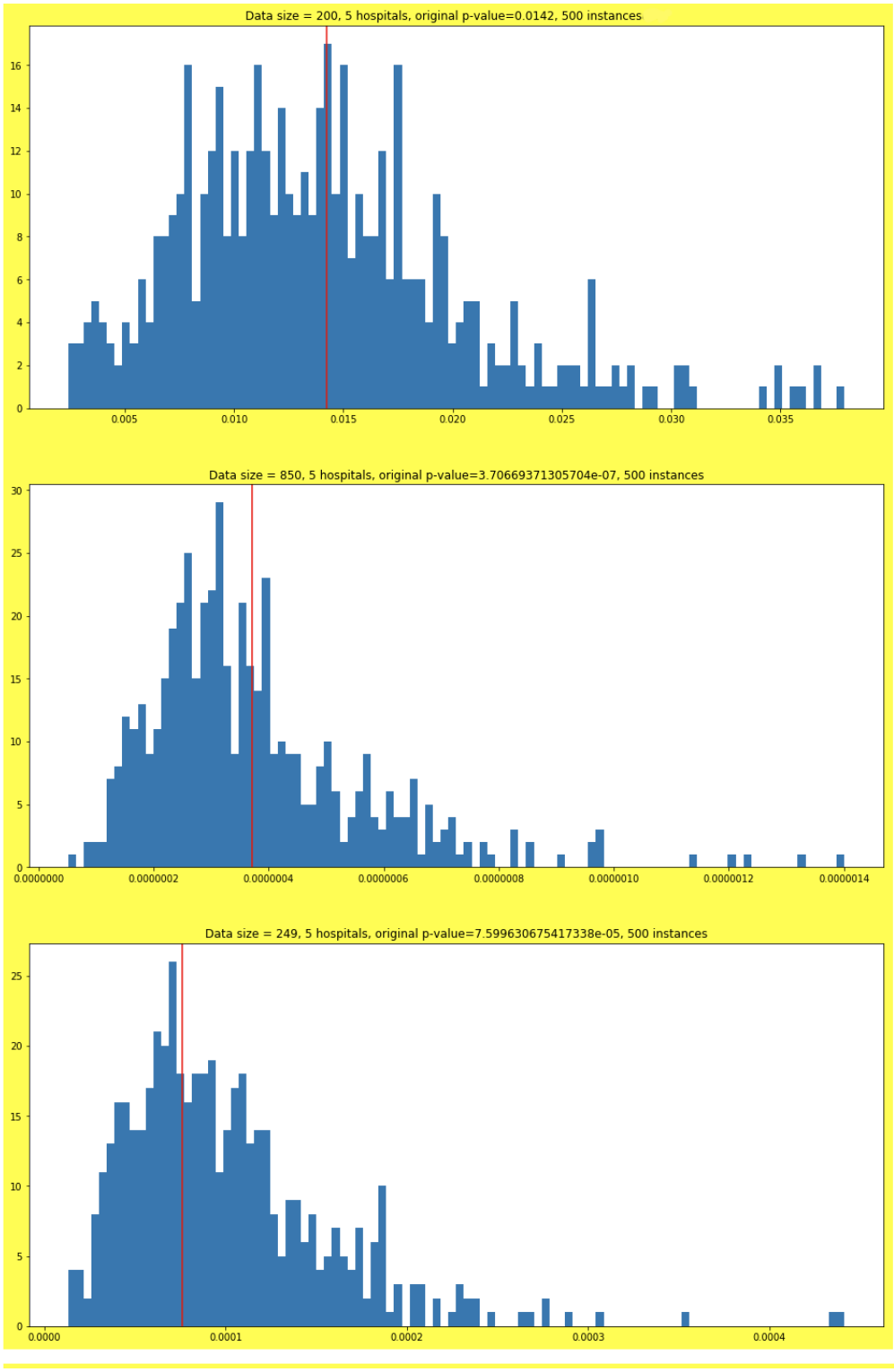


Figure 4.1: The same simulation over 3 datasets. Different patients allocation results in different p-values. See text for details.

$Z^*$ was calculated assuming each subset is owned by different party. A new p-value was calculated from this $Z^*$.

This process was repeated 500 times to generate 500 new p-values based on $Z^*$ values for 500 random partitions. The top histogram of Figure 4.1 consists of the 500 p-value results.

In the middle histogram we can see the same simulation performed on a different dataset of 850 patients generating a p-value of 3.7066e-07 (marked in red). The bottom histogram presents the same simulation with a third dataset or 249 patients. The Logrank p-value is 7.6e-05 (in red). The three datasets are taken from different breast cancer cohorts - MAIZ[19], UPSA[2] and STAM[4].

Observing the histograms we can clearly see that evidently the proposed protocol doesn't comply with the standard Logrank test: $Z \neq Z^*$. The underlying reason for this is a loss of information: the sorting procedure in standard Logrank Test holds larger amount of information (timing comparisons) compared to the partial sorting procedures performed by each party locally on smaller portions of the original dataset. At first glance it may look like a major drawback. However, we will argue that $Z^*$ is an alternative approach which also represents adequate significance testing for survival analysis.

In particular, we will take two steps:

1. Proving that $Z^* \sim N(0,1)$ (see section 4.3)

   The reliability of the p-value calculation in CoPPSA is not based on the correctness of the Logrank test, but rather on the strong version of the Central Limit Theorem. We will prove that, assuming a sufficiently large number of subjects, $Z^*$ has an approximately $N(0,1)$ distribution under the null model, which presumes that the survival function is independent of the class.

2. Empirically demonstrating that with high probability $|Z^* - Z|$ is small.

   As can be observed in the histograms 4.4-4.11, the p-values calculated from $Z$ and $Z^*$ are relatively close. In section 4.4 we will investigate the distance between $Z$ and $Z^*$.

## 4.2 The Difference Between $Z$ And $Z^*$

For further intuition about the difference between $Z$ and $Z^*$, consider the following synthetic example.

Consider a case in which two hospitals, Hospital1 and Hospital2, wish to measure the efficiency of new blue medicine called "The Blue Cure". The two hospitals started the following experiment at the exact same time, separately. Each hospital collects a group of volunteers within the patients and divides them to two groups. One group is given "The Blue Cure" and the other is given a placebo. Each hospital measures the time of patients' death (note that in this example all events are failures, that is: no censorship). Eventually, the two hospitals merge the collected data into one timeline of failures.

The merged timeline is presented in Figure 4.2. A blue dot denotes a death of a patient that was given "The Blue Cure". A blank dot denotes a death of a patient that was given a placebo.

Figure 4.2: The merged data from both hospitals. The blank dots denote failures of patients from the placebo group. The blue dots denotes failures of patients from "The Blue Cure" group.

We can see that the first three patients to fail were given a placebo, and the five patients who survived the longest were given "The Blue Cure". Note that we cannot tell the association between patients and hospitals.

Observing the timeline, we see that the patients that were given "The Blue Cure" tended to live longer. Without any calculations we can clearly deduce $Z$ is negative if the target group is the "blue" group (that was given the blue medicine).

$$O = \sum_{j=1}^{J} o_{blue,j} \quad E = \sum_{j=1}^{J} E_{blue,j} \quad V = \sum_{j=1}^{J} V_{blue,j}$$

$$Z_{blue} = \frac{O - E}{\sqrt{V}} < 0$$

Let $Z_0$ be Logrank Test result as calculated from this timeline. $Z_0 = Z_{blue} < 0$ . Our next step is to show that for different allocations to subsets, $Z^*$ changes while $Z_0$ is the same.

What would have happened if the dataset collected in each hospitals weren't merged? In Figure 4.3 we can see two distinct patient allocations for the same merged timeline as presented in Figure 4.2. The two allocations are labeled "Division A" and "Division B". The timelines in Figure 4.2 labeled "Merged Data" denotes the merged timeline, and they are identical, as expected. The purple square denotes the patients allocated for Hospital 2 in division A. They appear also on the separate timeline labeled "Hospital2" under "Division A". The orange square denotes the patients allocated for Hospital 2 in division B. They appear also on the separate timeline labeled "Hospital2" under "Division B".
In both allocations Hospital2 dataset consist of 6 patients, 3 of which are given "The Blue Cure" and 3 in are given placebo.

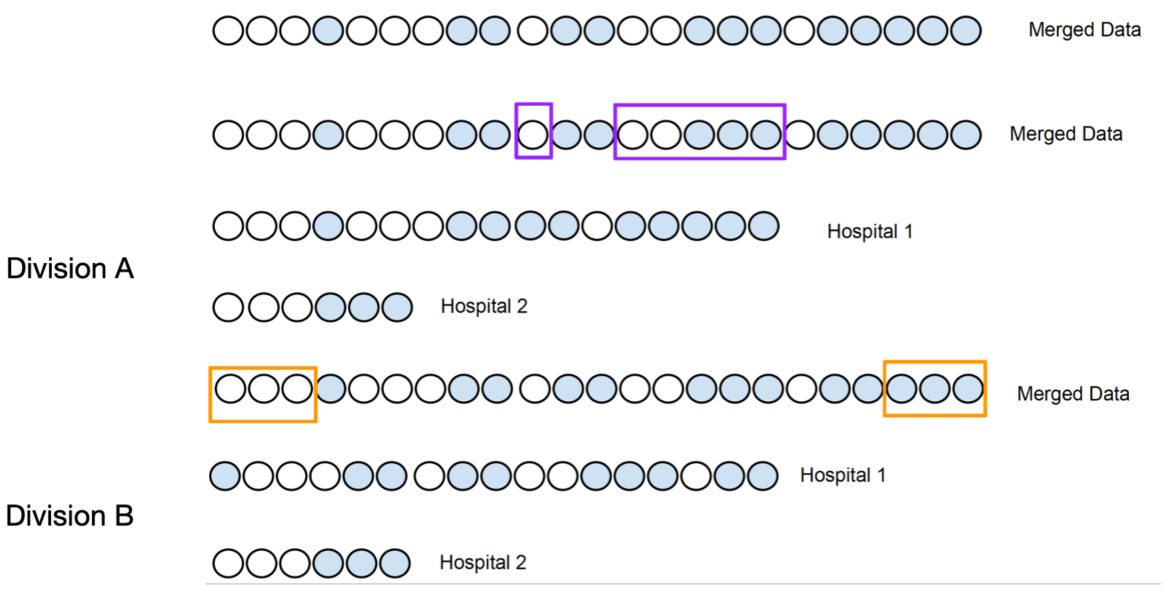


Figure 4.3: Two distinct patient allocations for the same merged timeline results in two different values of $Z^*$

The CoPPSA result based on Division A is denoted by $Z^*_A$, and the CoPPSA result based

on Division B is denoted by $Z_B^*$. The standard Logrank Test result is equal to $Z_0 < 0$ for all possible divisions, because they are all merged to the same identical dataset.

Observations:

1. In both allocations Hospital2 has the same timeline, and therefore the same values $O(2)_A = O(2)_B, E(2)_A = E(2)_B, V(2)_A = V(2)_B$.

2. Hospital1 has different timelines, so the values are different: $\Big(O(1), E(1), V(1)\Big)_A \neq \Big(O(1), E(1), V(1)\Big)_B$.

From the observations we can therefore conclude that $Z_A^* \neq Z_B^*$. Therefore: $Z_0 \neq Z_A^*$ or $Z_0 \neq Z_B^*$.

Conclusion:

The protocol calculation result $Z^*$ is not necessarily equal to the Logrank result $Z$.

## 4.3 CoPPSA - Proof Of Convergence to Standard Normal Distribution

### 4.3.1 Intuition

To validate the correctness of the protocol we need to prove the following claim:

$Z^* = \frac{\sum_{i=1}^{N} O(i) - E(i)}{\sqrt{\sum_{i=1}^{N} V(i)}}$ Then under the null model: $Z^* \sim N(0,1)$.

Let $(O_{A,1}, ..., O_{A,k})$ be the set of random variables defined in the Logrank process over the

dataset D of some party.

$$O_{A,j} \sim HG(n_j, n_{A,j}, o_j)$$

The observed value of each random variable $O_j$ is $o_j$.

The expected value and the variance of $O_{A,j}$ under the null model:

$$E_{A,j} = \frac{n_{A,j}}{n_j} o_j, \quad V_{A,j} = \frac{n_{A,j}}{n_j} o_j \Big(\frac{n_j - o_j}{n_j}\Big)\Big(\frac{n_j - n_{A,j}}{n_j - 1}\Big)$$

Under the null model $(O_{A,1}, ..., O_{A,k})$ are collectively independent. By the Logrank process we define:

$$O = \sum_{j=1}^{k} O_{A,j} \quad E = \sum_{j=1}^{k} E_{A,j} \quad V = \sum_{j=1}^{k} V_{A,j}$$

Assuming that $k$ is large enough we get the Gaussian $(O - E)$ with the variance $V$

Before we describe the proof, Let's try an intuitive (informal) approach and extend the above to a case with multiple datasets.

$\{D(1), D(2), ...D(N)\}$ is a set of N datasets of N parties respectively.

$\Big\{O_{A,1}(i), O_{A,2}(i), ..., O_{A,k_i}(i)\Big\}$ are the random variables defined from dataset D(i) in the Logrank process.

$$O(i) = \sum_{j=1}^{k} O_{A,j}(i) \quad E(i) = \sum_{j=1}^{k} E_{A,j}(i) \quad V(i) = \sum_{j=1}^{k} V_{A,j}(i)$$

Consider the set of all random variables from all datasets:

$$\Big\{O_{A,1}(1), O_{A,2}(1), ..., O_{A,k_1}(1), ..., O_{A,1}(N), O_{A,2}(N), ..., O_{A,k_N}(N)\Big\}$$

Under the null model all these RVs are collectively independent. Therefore, the Gaussian random variables $\forall i = 1, ...N : (O(i) - E(i))$ are also collectively independent.

Therefore, $\sum_{i=1}^{N}(O(i)-E(i))$ is a Gaussian with variance $\sum_{i=1}^{N}V(i)$

By this we get that $Z^* = \frac{\sum_{i=1}^{N}(O(i)-E(i))}{\sqrt{\sum_{i=1}^{N}(V(i))}} \sim N(0,1)$

In the next sections we will give the complete proof of the convergence.

### 4.3.2 Preliminary Definitions And Claims

**Theorem 4.3.1.** *(**Lindeberg-Feller Theorem**)*

*Let $(Y_1, ..., Y_n)$ be a set of independent RVs s.t. $\forall i \in [1,n] : E(Y_i) = 0$.*

- $T_n = \sum_{k=1}^{n} Y_k$
- $Var(Y_k) = \sigma_k^2$
- $s_n^2 = Var(T_n) = \sum_{k=1}^{n} Var(Y_k) = \sum_{k=1}^{n} \sigma_k^2$

*If for every $\epsilon > 0$ $\lim_{n\to\infty} \frac{1}{s_n^2} \sum_{k=1}^{n} E\Big(Y_k^2 \cdot \mathbb{I}\{|Y_k| \geq \epsilon \cdot s_n\}\Big) = 0$*

*then: $\frac{T_n}{s_n} \xrightarrow{D} N(0,1)$*

The Condition $\forall \epsilon > 0$ $\lim_{n\to\infty} \frac{1}{s_n^2} \sum_{k=1}^{n} E\Big(Y_k^2 \cdot \mathbb{I}\{|Y_k| \geq \epsilon \cdot s_n\}\Big) = 0$ is called **Lindeberg's Condition**.

**Lemma 4.3.2.** *Let $(Y_1, ..., Y_n)$ be a set of discrete independent random variables s.t.*

1. *$\forall i \in [1,n] : -M \leq Y_i \leq M$ for some constant $M > 0$*
2. *$\forall i \in [1,n] : E(Y_i) = 0$*
3. *$\forall i \in [1,n] : Var(Y_i) > 0$*
4. *$\lim_{n\to\infty} \sum_{i=1}^{n} Var(Y_i) = \infty$*

*Then: $(Y_1, ..., Y_n)$ complies with Lindeberg's condition.*

*Proof.* Notations:

- $T_n = \sum_{k=1}^{n} Y_k$

- $Var(Y_k) = \sigma_k^2$
- $s_n^2 = Var(T_n) = \sum_{k=1}^{n} Var(Y_k) = \sum_{k=1}^{n} \sigma_k^2$

Now, we show that Lindeberg condition holds:

$$\frac{1}{s_n^2}\sum_{k=1}^{n} E(Y_k^2 \cdot \mathbb{I}\{|Y_k| \geq \epsilon \cdot s_n\}) = \frac{1}{s_n^2}\sum_{k=1}^{n}\sum_{y=-M}^{M} P(Y_k = y)y^2 \cdot \mathbb{I}\{|y| \geq \epsilon \cdot s_n\}dy \leq$$

$$\leq \frac{1}{s_n^2}\sum_{k=1}^{n} M^2 \sum_{y=-M}^{M} P(Y_k = y) \cdot \mathbb{I}\{|y| \geq \epsilon \cdot s_n\}dy = \frac{1}{s_n^2}\sum_{k=1}^{n} M^2 P(|Y_k| \geq \epsilon \cdot s_n) \leq$$

$$\xrightarrow{Chebyshev's, \lambda = \epsilon \cdot s_n} \leq \frac{1}{s_n^2}\sum_{k=1}^{n} M^2\Big(\frac{\sigma_k^2}{\epsilon^2 s_n^2}\Big) = \frac{M^2}{s_n^2} \cdot \frac{1}{\epsilon^2 s_n^2}\sum_{k=1}^{n}\sigma_k^2 = \frac{M^2 s_n^2}{s_n^2 \epsilon^2 s_n^2} = \frac{M^2}{s_n^2 \epsilon^2}$$

$\lim_{n\to\infty}\sum_{i=1}^{n} Var(Y_i) = \lim_{n\to\infty} s_n^2 = \infty$ , so for every $\epsilon > 0$ :

$$\lim_{n\to\infty} \frac{M^2}{s_n^2 \epsilon^2} = 0$$

$$0 \leq \frac{1}{s_n^2}\sum_{k=1}^{n} E(Y_k^2 \cdot \mathbb{I}\{|Y_k| \geq \epsilon \cdot s_n\}) \leq \frac{M^2}{s_n^2 \epsilon^2}$$

Therefore, for every $\epsilon > 0$ :

$$\lim_{n\to\infty} \frac{1}{s_n^2}\sum_{k=1}^{n} E\Big(Y_k^2 \cdot \mathbb{I}\{|Y_k| \geq \epsilon \cdot s_n\}\Big) = 0$$

Conclusion: $(Y_1, ..., Y_n)$ complies with Lindeberg condition. □

**Lemma 4.3.3.** *Let D be a Logrank dataset. Let* $\Big(O_{A,1}, O_{A,2}, ..., O_{A,k}\Big)$ *be the set of random variables defined in the Logrank process over the dataset D.*
*then:* $\lim_{k\to\infty}\sum_{j=1}^{k} Var(O_{A,j}) = \infty$

### 4.3.3 Convergence To Standard Normal Distribution

In this section we use theorem 4.3.1, lemma 4.3.2 and lemma 4.3.3 to prove that $Z^* \sim N(0, 1)$.

Let $D(1), D(2), ...D(N)$ be a set of survival time-and-event datasets used by CoPPSA.

Define the set of new random variables $(Y_{11}, Y_{21}, ...., Y_{n_N,N})$ s.t. For each i,j $Y_{ji} = O_{A,j}(i) - E_{A,j}(i)$.

**Claim 4.3.4.** $(Y_{11}, Y_{21}, ...., Y_{n_N,N})$ *is a set of discrete RVs (trivial)*

**Claim 4.3.5.** $(Y_{11}, Y_{21}, ...., Y_{n_N,N})$ *is a set of collectively independent RVs.*

*Proof.* Each party $i = 1, ..., N$ calculates the vector $\Big(o_j(i), E_{A,j}(i), V_{A,j}(i)\Big)$ for each time interval j. Just like in the standard Logrank, $O_{A,j}(i)$ is a random variable with hyper-geometric distribution. And similarly to Logrank $(O_{A,1}(i), O_{A,2}(i), ..., O_{A,T_i}(i))$ is collectively independent. Since each party conducts a separate independent experiment, we can assume that

$$\Big(O_{A,1}(1), O_{A,2}(1), ..., O_{A,k_1}(1), ..., O_{A,1}(N), O_{A,2}(N), ..., O_{A,k_N}(N)\Big)$$

are collectively independent. Therefore $(Y_{11}, Y_{21}, ...., Y_{n_N,N})$ are also collectively independent.

□

**Claim 4.3.6.** $-1 \leq Y_{ji} \leq 1$

*Proof.* In both standard Logrank Test and CoPPSA, each failure time defines a new time interval. $O_{A,j}(i) > 1$ meas that several failures occurred at the exact same time. In a theoretic, experiment in which time measurement resolution is infinite, we get:

$$\forall i,j : 0 \leq O_{A,j}(i) \leq 1 \Longrightarrow 0 \leq E_{A,j}(i) \leq 1 \Longrightarrow -1 \leq O_{A,j}(i) - E_{A,j}(i) \leq 1 \Longrightarrow -1 \leq Y_{ji} \leq 1$$

(In practice, we can tune the time measurement resolution of the experiment according to an arbitrary M, and say $O_{A,j}(i) \leq M$). □

**Claim 4.3.7.** $E(Y_{ji}) = 0$ *(trivial)*

**Claim 4.3.8.** $Var(Y_{ji}) > 0$

*Proof.* By the definition of Hyper-geometric distribution we know:
$Var(Y_j) = V_{A,j} = \frac{n_{A,j}}{n_j} o_j \Big(\frac{n_j - o_j}{n_j}\Big)\Big(\frac{n_j - n_{A,j}}{n_j - 1}\Big)$
By definition: $\forall i,j : 0 < n_{A,j}(i) \quad o_j(i) < n_j(i) \quad 2 \leq n_j(i) \quad \Longrightarrow Var(Y_{ji}) > 0$ □

**Claim 4.3.9.** $\lim_{n\to\infty}\sum_{i=1}^{N}\sum_{j=1}^{n} Var(Y_{ji}) = \infty$

*Proof.* From Lemma 4.3.3 we assume: $\lim_{T_i\to\infty}\sum_{j=1}^{T_i} Var(O_{A,j}(i)) = \infty$

$$\implies \lim_{T_i\to\infty}\sum_{j=1}^{T_i} Var(Y_{ji}) = \infty \implies \sum_{i=1}^{N}\lim_{T_i\to\infty}\sum_{j=1}^{T_i} Var(Y_{ji}) = \infty \implies \lim_{n\to\infty}\sum_{i=1}^{N}\sum_{j=1}^{n} Var(Y_{ji}) = \infty$$

We see that $(Y_{11}, Y_{21}, ...., Y_{n_N,N})$ is set of discrete independent variable s.t.

1. $\forall i,j : -1 \le Y_{ji} \le 1$
2. $\forall i,j : E(Y_{ji}) = 0$
3. $\forall i,j : Var(Y_{ji}) > 0$
4. $\lim_{n\to\infty}\sum_{i=1}^{N}\sum_{j=1}^{n} Var(Y_{ji}) = \infty$

So by lemma 4.3.2 and theorem 4.3.1:

$$\lim_{n\to\infty}\frac{\sum_{i=1}^{N}\sum_{j=1}^{T_i} Y_{ji}}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{T_i} Var(Y_{ji})}} \sim N(0,1)$$

By the original definition of $Y_{ji}$ :

$$\sum_{j=1}^{T_i} Var(Y_{ji}) = \sum_{j=1}^{T_i} Var(O_{A,j}(i)) = V(i), \quad \sum_{j=1}^{T_i} Y_{ji} = (O(i) - E(i)) \implies \lim_{n\to\infty}\frac{\sum_{i=1}^{N} O(i) - E(i)}{\sqrt{\sum_{i=1}^{N} V(i)}} \sim N(0,1)$$

□

## 4.4 How Close Are $Z$ And $Z^*$ ?

To justify the usage of the newly proposed algorithm we must show that $Z$ and $Z^*$ are close with high probability as the number of subjects increases and under some assumption related to partitioning the cohort of subjects. We defer a formal proof to future work (see discussion) and herein provide some empirical evidence. The histogram in Figure 4.4 is based

on the data from the MAINZ cohort [19]. This dataset has 200 entries, corresponding to 200 patients with breads cancer. As mentioned above, we divided the data according to cancer sub-type - Luminal A vs not Luminal A - and define the target group to be Luminal A. The standard Logrank calculation on this dataset results with a $Z = -2.19$.

The 200 values dataset was then uniformly randomly split into 5 subsets. Then $Z^*$ was calculated assuming each subset is owned by different party.

This process was repeated 500 times to generate 500 new $Z^*$ values for 500 random partitions. The histogram in Figure 4.4 consists of the 400 values of $Z^* - Z$.

In the following histograms, shown from Figure 4.5 up to Figure 4.11 we can see the output of the same simulation taken over different datasets (UPSA[2], STAM[4] and SCANB[3], different merges between them and with additional synthetic data). Each histogram consists of the 500 results of the calculation $Z^* - Z$, according to 500 random partitions. The comparison is between Luminal A and not Luminal A. The target group is always Luminal A.

Let $S(D)$ be the size of the dataset $D$. The shapes of histograms are close to a normal distribution shape. This indicates that $Z^* - Z$ may have a normal distribution such as $(Z - Z^*) \sim Normal\Big(0, V\big(S(D)\big)\Big)$ for an unknown function $V()$. An interesting pattern emerges: the larger the dataset is, the smaller the variance of the Gaussian becomes. Let $s$ be the size of the merged dataset, the pattern may indicate that: $\lim_{s\to\infty} V(s) = 0$ and therefore $\lim_{s\to\infty} Z^* = Z$

Assuming that this conjecture is true, a generalization may have the following form:

**Conjecture 4.4.1.** *for every $k$, $\epsilon$ and $\delta$ there exists an $N$ such that for every dataset of size $n$ such that $N \leq n$, if the dataset is randomly and uniformly divided into $k$ subsets with similar sizes, then:*

$$P(|Z - Z^*| > \epsilon) < \delta$$

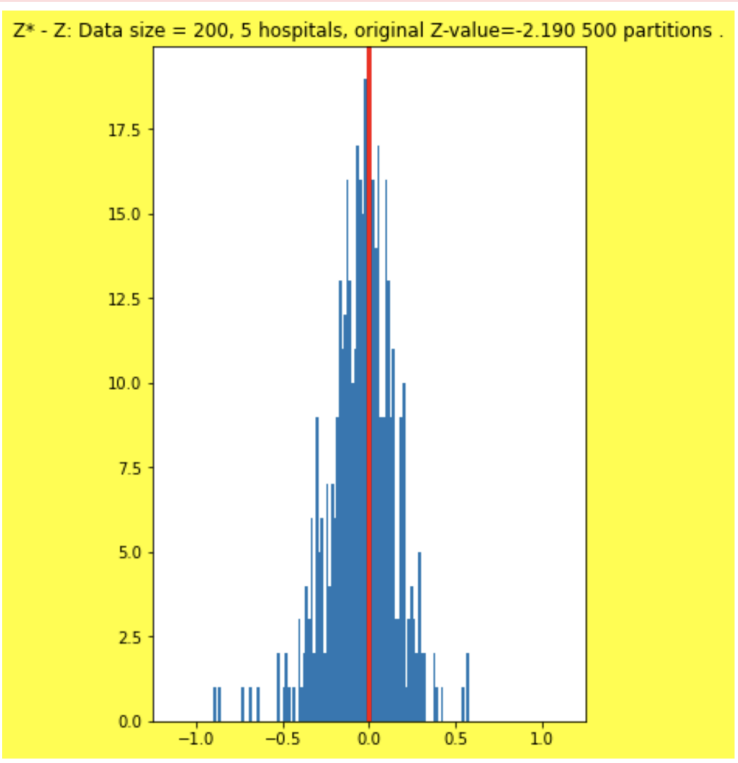


Figure 4.4: $Z^* - Z$ values for different partitions, 200 subjects

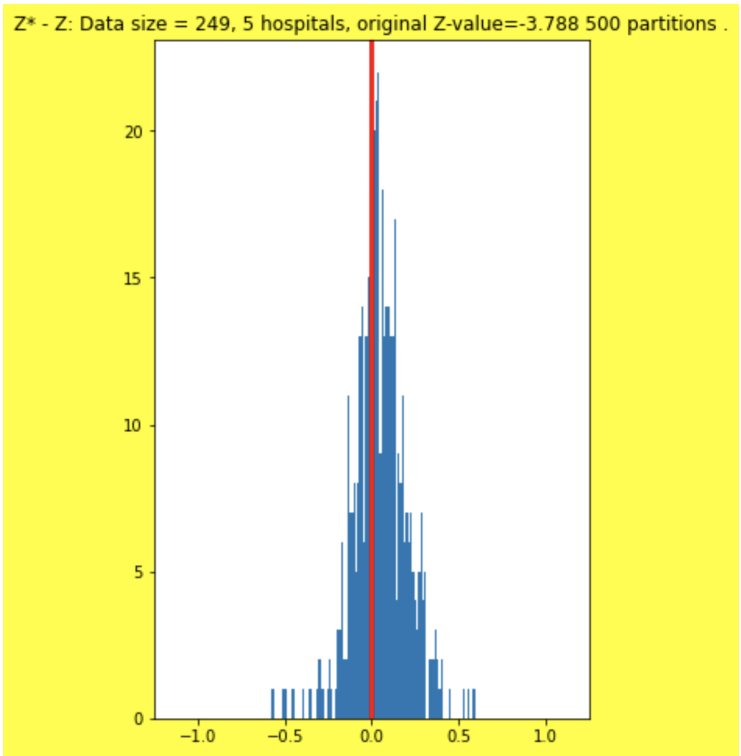


Figure 4.5: $Z^* - Z$ values for different partitions, 249 subjects

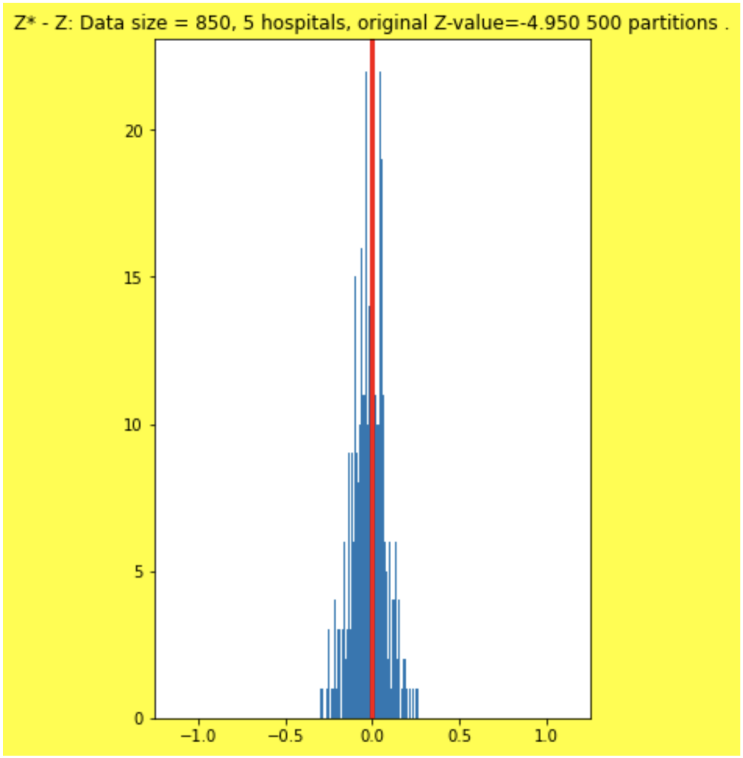


Figure 4.6: $Z^* - Z$ values for different partitions, 850 subjects

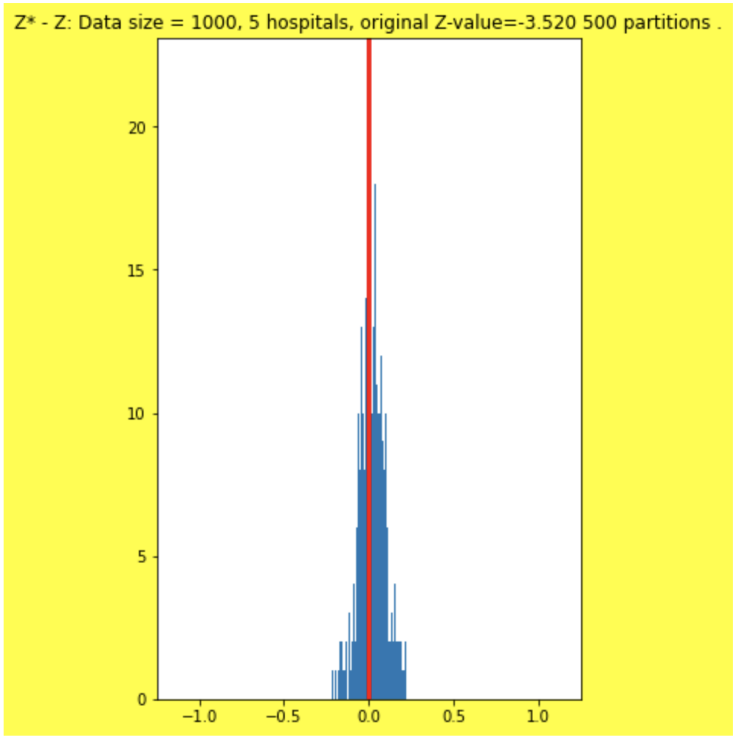


Figure 4.7: $Z^* - Z$ values for different partitions, 1000 subjects

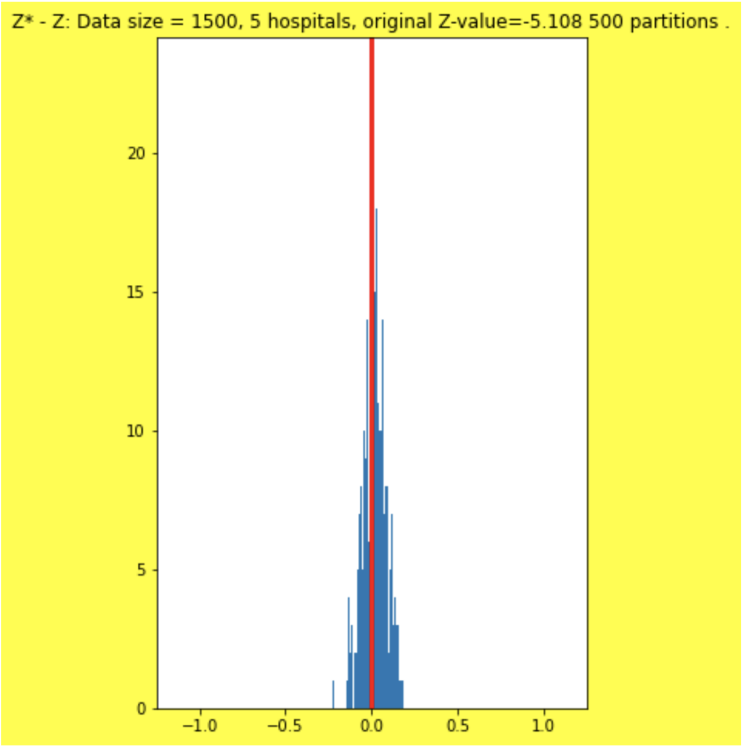


Figure 4.8: $Z^* - Z$ values for different partitions, 1500 subjects

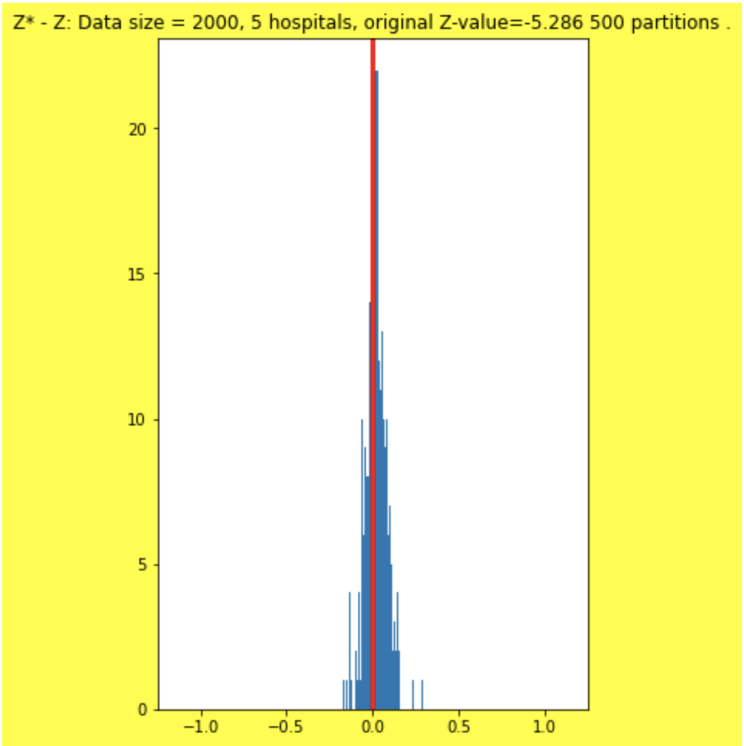


Figure 4.9: $Z^* - Z$ values for different partitions, 2000 subjects

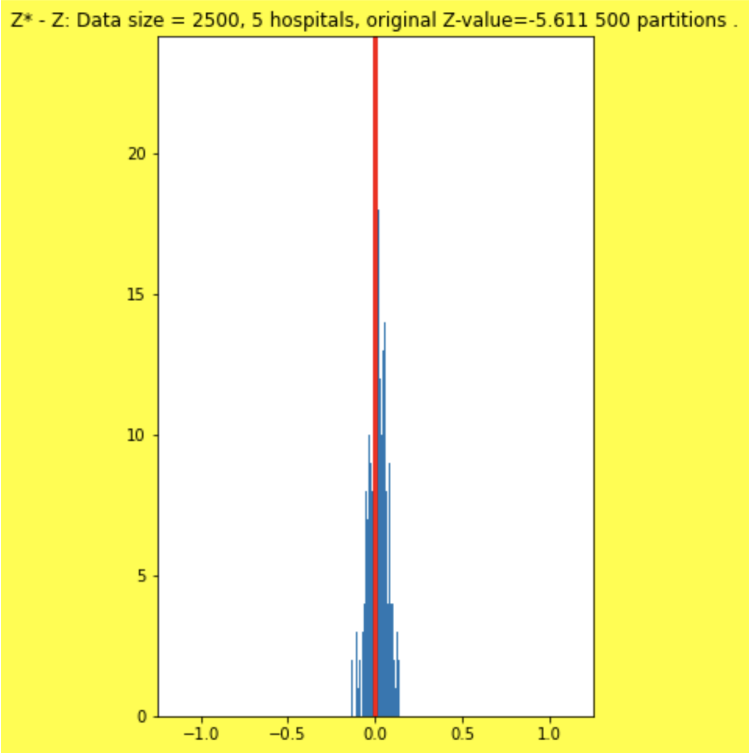


Figure 4.10: $Z^* - Z$ values for different partitions, 2500 subjects

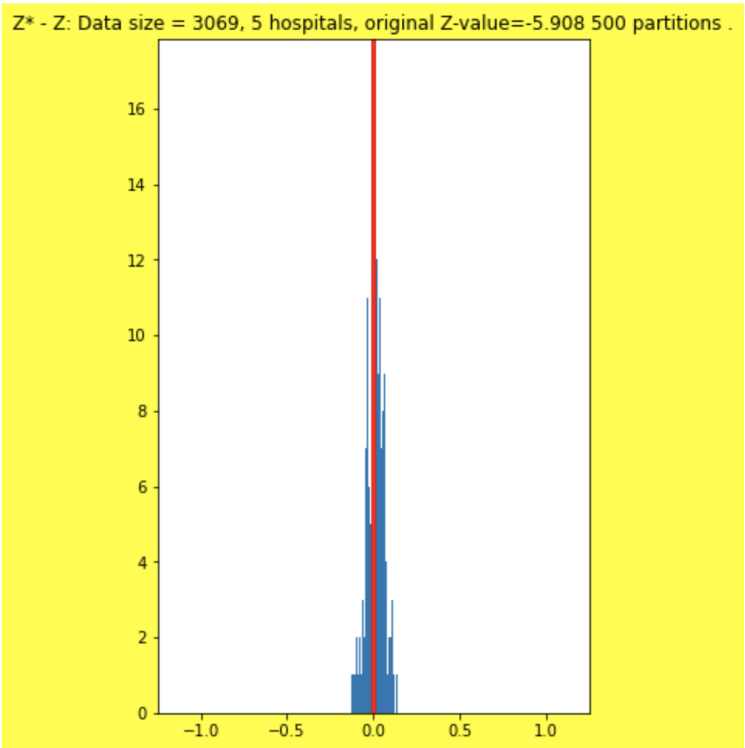


Figure 4.11: $Z^* - Z$ values for different partitions, 3069 subjects

# Chapter 5

# Incentive For Veracity

## 5.1 What Is A lie?

In the MPC field a false input is not considered as an attack. A secure MPC protocol is not necessarily protected against deceptions.

For example, let $\pi$ be a secure MPC protocol for calculating the function

$$f(x_1, x_2, x_3) = x_1 + x_2 + x_3$$

such that each party $P_i$ owns the input $x_i$.

The protocol output $f(x_1, x_2, x_3)$ is sent back to all three parties. If $P_1$ provides a false input $z_1 \neq x_1$ then the output of $\pi$ will be $f(z_1, x_2, x_3) = z_1 + x_2 + x_3$. By the end of the run $P_2$ and $P_3$ have the value $f(z_1, x_2, x_3)$, meaning they got a false output. But $P_1$ can easily calculate the desired and true value $f(x_1, x_2, x_3)$:

$$f(z_1, x_2, x_3) - z_1 + x_1 = z_1 + x_2 + x_3 - z_1 + x_1 = f(x_1, x_2, x_3)$$

Now, let's consider a similar scenario in CoPPSA:

Party $P_l$ "lies" (i.e. provides a false input), while all other parties use their true results

(see section 3.1).

The true input of $P_l$:

$$m_{1_{true}} = \lfloor 10^b \cdot (O(l) - E(l)) \rfloor, \quad m_{2_{true}} = \lfloor 10^b \cdot V(l) \rfloor$$

$P_l$ maliciously uses fake input.

$$M_{fake} = (m_{1_{fake}} \mod p, m_{2_{fake}} \mod p)$$

All parties receive the result

$$D_{wrong} = (\sum_{i=1}^{N} m_1(i)) \, mod \, p$$

$$U_{wrong} = (\sum_{i=1}^{N} m_2(i)) \, mod \, p$$

$P_l$ calculates

$$D_{fixed} = \left( (\sum_{i=1}^{N} m_1(i)) - m_{1_{fake}} + m_{1_{true}} \right) \mod p = (\sum_{i=1}^{N} m_1(i)) - m_{1_{fake}} + m_{1_{true}}$$

$$U_{fixed} = \left( (\sum_{i=1}^{N} m_2(i)) - m_{2_{fake}} + m_{2_{true}} \right) \mod p = (\sum_{i=1}^{N} m_2(i)) - m_{2_{fake}} + m_{2_{true}}$$

Party $l$ calculates the true result $Z^* = \frac{D_{fixed} \cdot 10^{-b}}{\sqrt{U_{fixed} \cdot 10^{-b}}}$

Eventually $P_l$ is the only party who knows the correct $Z^*$ value.

### 5.1.1 Deterministically Non-Cooperatively Computable Functions

Y. Shoham and M. Tennenholtz presented in 2005 the concept of deterministically non-cooperatively computable functions (D-NCC)[20]. They defined a boolean function to be

D-NCC as follows:

Assume a set of n agents (parties) $\{1, 2, ..., n\}$ and a special agent termed 'the center'. For each agent $i$: $v_i \in \{0,1\}$ is the type (input) of agent $i$. The vector of agent types $v = (v_1, ..., v_n)$ has a joint probability distribution $P$. $P$ assigns positive probability to all configurations, i.e. $P(v) > 0$ for every $v \in \{0,1\}^n$.

Given a function $w : \{0,1\}^n \to \{0,1\}$, we consider the following protocol:

1. For any vector $v \in \{0,1\}^n$, each agent $i$ declares his type $\tilde{v_i}$ to the center. It can be either true ($\tilde{v_i} = v_i$) or false ($\tilde{v_i} \neq v_i$)

2. The center computes $w(\tilde{v}) = w(\tilde{v_1}, ..., \tilde{v_n})$ and send the value $w(\tilde{v})$ to all agents.

3. Each agent $i$ computes $w(v)$ based on $w(\tilde{v})$ and $v_i$ (his true input).

A pair of functions $(f_i, g_i)$ represents the strategy for agent $i$ .

- $f_i : \{0,1\} \to \{0,1\}$ is the **declaration function**. It sets the input sent to the center based on the true input. $f_i(v_i) = \tilde{v_i}$

- $g_i : \{0,1\}^2 \to \{0,1\}$ is the **interpretation function**. It is used by the agent to "fix" the output based on the center's announcement and his own true input. $g_i(w(\tilde{v}), v_i)$

Notations:

- $v_{-i} = (v_1, v_2, ..., v_{i-1}, v_{i+1}, ...v_n)$

- For $b \in \{0,1\}$: $(b, v_{-i}) = (v_1, ..., v_{i-1}, b, v_{i+1}, ...v_n)$

**Definition 5.1.1.** *Let $n, P, w$ be as defined above . Then $w$ is deterministically non-cooperatively computable (D-NCC) if:*
*For any agent $i$, every strategy $(f_i, g_i)$ and every $v_i \in \{0,1\}$, the following holds.*
*Either there exists $v_{-i} \in \{0,1\}^{n-1}$ such that $g_i(w(f_i(v_i), v_{-i}), v_i) \neq w(v_i, v_{-i})$,*
*or else for every $v_{-i} \in \{0,1\}^{n-1}$ we have $w(f_i(v_i), v_{-i}) = w(v_i, v_{-i})$.*

Simply put, if an agent can impact the center's output value, then for every false value it produces, there are types of the other agents, that will result in an output he can not fix.

The definition is powerful and intuitive, but it applies for boolean functions only, and it also assumes $P(v) > 0$ for each $v$.

For our purpose - incentive for honesty in CoPPSA - we propose our own new definition. This definition is very similar to the definition of D-NCC, but it also holds for non-boolean functions and inputs.

## 5.2 What Is An Incentive For Good Veracity?

We would like to construct a protocol that motivates the parties to provide true input values. We call this feature "Incentive for good behavior" or "Incentive for veracity".

Before we start, let's recall the definition of conditional entropy.
Consider two random variables $X, Y$ with images $\mathfrak{X}, \mathfrak{Y}$ respectively.
For discrete variables the conditional entropy is defined as:

$$H(Y|X) = -\sum_{x \in \mathfrak{X}} P(X = x) \sum_{y \in \mathfrak{Y}} P(Y = y|X = x) \log P(Y = y|X = x)$$

**Definition 5.2.1.** *(A function with an incentive for veracity with respect to one input)*
*Let* $\{X_1, ..., X_n\}$ *be a set of independent random variables,* $\forall i \in \{1, .., n\} \quad X_i : \Omega \to \mathbb{R}$.
*Let* $f(x_1, ..., x_n)$ *be a function* $f : \mathbb{R}^n \to \mathbb{F}$.
*Consider* $l \in \{1, ..., n\}$. *We say* $f(X_1, ..., X_n)$ ***has an incentive for veracity with respect to input*** $X_l$ *if for every couple of sample values* $\tau, \varphi \in \mathbb{R}$ *s.t.* $P(X_l = \tau) > 0, P(X_l = \varphi) > 0$ *and* $\tau \neq \varphi$*:*

$$H(f(X_1, ..., X_l = \tau, ..., X_n)|f(X_1, ..., X_l = \varphi, ..., X_n)) > 0$$

*Where $H(\cdot|\cdot)$ denotes the conditional entropy and the probability is taken over the distribution of the inputs $(X_1, ..., X_{l-1}, X_{l+1}, ..., X_n)$.*

**Claim 5.2.2.** *Let $g : \mathbb{F} \to \mathbb{F}$ be an injective function. $f : \mathbb{R}^n \to \mathbb{F}$. Let $\{X_1, ..., X_n\}$ be a set of collectively independent random variables.*
*If $f(X_1, ..., X_n)$ has an incentive for veracity with respect to $X_l$ then $g \circ f(X_1, ..., X_n)$ has an incentive for veracity with respect to $X_l$.*

*Proof.*

$$H(f(X_1, ..., X_l = \tau, ..., X_n)|f(X_1, ..., X_l = \varphi, ..., X_n)) > 0 \to$$

$$\to H(f(X_1, ..., X_l = \tau, ..., X_n)|g \circ f(X_1, ..., X_l = \varphi, ..., X_n)) > 0 \to$$

$$\to H(g \circ f(X_1, ..., X_l = \tau, ..., X_n)|g \circ f(X_1, ..., X_l = \varphi, ..., X_n)) > 0$$

□

**Definition 5.2.3.** *(A function with an incentive for veracity )*
*Let $\{X_1, ..., X_n\}$ be a set of independent random variables, $\forall i \in \{1, .., n\} \quad X_i : \Omega \to \mathbb{R}$.*
*Let $f(x_1, ..., x_n)$ be a function $f : \mathbb{R}^n \to \mathbb{F}$.*

*We say $f(X_1, ..., X_n)$* ***has an incentive for veracity*** *if $f(X_1, ..., X_n)$ has an incentive for veracity with respect to input $X_l$ for every $l \in \{1, ..., n\}$.*

**Example:**

Let $X_1, X_2, X_3$ be iid random variables, $X_i \in \{-1, 1\}$ , $P(X_i = 1) = P(X_i = -1) = 0.5$
$f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3x_1$

**Claim 5.2.4.** *$f(X_1, X_2, X_3)$ has an incentive for veracity.*

*Proof.* Notice that $f(X_1, X_2, X_3) = \begin{cases} 3 & \text{if } X_1 = X_2 = X_3 \\ -1 & else \end{cases}$

If $a = 1$ and $f(a, X_2, X_3) = 3$ then we know that $X_2 = 1$ and $X_3 = 1$ and we can calculate $f(-1, X_2, X_3) = -1$. But if $a = 1$ and $f(a, X_2, X_3) = -1$ then $X_2$ and $X_3$ are unknown and $f(-1, X_2, X_3)$ can be either $-1$ or 3.

In the same way, If $a = -1$ and $f(a, X_2, X_3) = 3$ then we know that $X_2 = -1$ and $X_3 = -1$ and we can calculate $f(1, X_2, X_3) = -1$. But if $a = -1$ and $f(a, X_2, X_3) = -1$ then $X_2$ and $X_3$ are unknown and $f(1, X_2, X_3)$ can be either $-1$ or 3.

Notations:
$f(1, X_2, X_3) = \tilde{f}_1(X_2, X_3)$ in short: $\tilde{f}_1()$
$f(-1, X_2, X_3) = \tilde{f_{-1}}(X_2, X_3)$ in short: $\tilde{f_{-1}}()$

Assume $a = -1, b = 1$.
$P(\tilde{f}_1() = -1 | \tilde{f_{-1}}() = 3) \log P(\tilde{f}_1() = -1 | \tilde{f_{-1}}() = 3) = 1 \cdot 0 = 0$
$P(\tilde{f}_1() = 3 | \tilde{f_{-1}}() = 3) \log P(\tilde{f}_1() = 3 | \tilde{f_{-1}}() = 3) = 0 \cdot -\infty = 0$
$P(\tilde{f}_1() = -1 | \tilde{f_{-1}}() = -1) \log P(\tilde{f}_1() = -1 | \tilde{f_{-1}}() = -1) = \frac{2}{3} \log \frac{2}{3}$
$P(\tilde{f}_1() = 3 | \tilde{f_{-1}}() = -1) \log P(\tilde{f}_1() = 3 | \tilde{f_{-1}}() = -1) = \frac{1}{3} \log \frac{1}{3}$

$$H\Big(\tilde{f}_1() | \tilde{f_{-1}}()\Big) = -\sum_{\tilde{f_{-1}}()=3,-1} P(\tilde{f_{-1}}()) \sum_{\tilde{f}_1()=3,-1} P(\tilde{f}_1() | \tilde{f_{-1}}()) \log P(\tilde{f}_1() | \tilde{f_{-1}}()) =$$

$$= -P\Big(\tilde{f_{-1}}() = 3\Big) \cdot 0 - P\Big(\tilde{f_{-1}}() = -1\Big) \cdot [\frac{2}{3} \log \frac{2}{3} + \frac{1}{3} \log \frac{1}{3}] = -\frac{3}{4}\Big(\frac{2}{3} \log \frac{2}{3} + \frac{1}{3} \log \frac{1}{3}\Big) > 0$$

The calculation is the same for $a = 1, b = -1$:

$$H\Big(\tilde{f_{-1}}() | \tilde{f}_1()\Big) = -\sum_{\tilde{f}_1()=3,-1} P(\tilde{f}_1()) \sum_{\tilde{f_{-1}}()=3,-1} P(\tilde{f_{-1}}() | \tilde{f}_1()) \log P(\tilde{f_{-1}}() | \tilde{f}_1()) > 0$$

We showed that $f$ has an incentive for veracity with respect to $X_1$. Due to the symmetry of the function, and since the the inputs are iid, $f$ has an incentive for veracity with respect to $X_2$ and $X_3$ .

Conclusion: $f$ has an incentive for veracity. □

**Example:** Let $X_1, X_2, X_3$ be random variables, $X_i \in \{1, 2\}$ , $P(X_i = 1) = P(X_i = 2) = 0.5$

$f(x_1, x_2, x_3) = \frac{x_1}{x_2} + \frac{x_2}{x_3} + \frac{x_3}{x_1}$

$f(X_1, X_2, X_3)$ has an incentive for veracity.

**Definition 5.2.5.** *(A function set with an incentive for veracity with respect to a set of inputs)*

*Let* $\{X_1, ..., X_n\}$ *be a set of independent random variables,* $\forall i \in \{1, .., n\} \quad X_i : \Omega \to \mathbb{R}$.

*Let* $f(x_1, ..., x_n)$ *be a function* $f : \mathbb{R}^n \to \mathbb{F}$.

*Consider* $(l_1, ..., l_k) \in \{1, ..., n\}$. *We say* $f(X_1, ..., X_n)$ ***has an incentive for veracity with respect to the inputs*** $(X_{l_1}, ..., X_{l_k})$ *if for every couple of vectors* $(\varphi_1, ..., \varphi_k), (\tau_1, ..., \tau_k) \in \mathbb{R}^k$ *s.t* $\forall i \in (1, ..., k) \quad P(X_{l_i} = \varphi_i) > 0, P(X_{l_i} = \tau_i) > 0$ *and* $(\varphi_1, ..., \varphi_k) \neq (\tau_1, ..., \tau_k)$*:*

$$H(f_l(X_1, ..., X_{l_1} = \tau_1, ..., X_{l_k} = \tau_k, ..., X_n) | f_l(X_1, ..., X_{l_1} = \varphi_1, ..., X_{l_k} = \varphi_k, ..., X_n)) > 0$$

*Where the probability is taken over the distribution of the inputs* $\{X_1, ..., X_n\}/\{X_{l_1}, ..., X_{l_k}\}$.

**Definition 5.2.6.** *(Protocol with incentive for veracity )*

*Let* $\{k_1, ..., k_n\}$ *be a set of positive integers.*

*Let* $\bar{x_1} = (x_{1,1}, ..., x_{1,k_1}), \bar{x_2} = (x_{2,1}, ..., x_{2,k_2}) ..., \bar{x_n} = (x_{n,1}, ..., x_{n,k_n})$ *be vectors of variables.*

*Let* $f(\bar{x_1}, \bar{x_2}, ..., \bar{x_n}) = \left(f_1(\bar{x_1}, ..., \bar{x_n},), f_2(\bar{x_1}, ..., \bar{x_n}), ..., f_n(\bar{x_1}, ..., \bar{x_n})\right)$

*be a function* $f : \mathbb{R}^{k_1 + ... + k_n} \to \mathbb{F}^n$ *that can be computed by a PPT algorithm.*

$\pi$ *is a secure MPC protocol for* $f(\bar{x_1}, \bar{x_2}, ..., \bar{x_n})$ *s.t* $\bar{x_l}$ *is the input vector of party* $P_l$ *and* $f_l(\bar{x_1}, ..., \bar{x_n})$ *is the output of party* $P_l$.

*Let* $\{\bar{X_1}, \bar{X_2}, ..., \bar{X_n}\}$ *be a set of independent random vectors,* $\bar{X_i} : \Omega \to \mathbb{R}^{k_i}$.

*Each random vector represents a set of secret inputs.*

*We say* ***protocol*** $\pi$ ***has an incentive for veracity*** *if for every* $l \in \{1, ..., n\}$*:*

$f_l(\bar{X_1}, ..., \bar{X_n})$ *has an incentive for veracity with respect to the inputs* $\bar{X_l} = (X_{l,1}, ..., X_{l,k_l})$.

## 5.3 How To Add The Incentive To CoPPSA?

To add incentive to CoPPSA we need to change the value that is being calculated in phase 2 of the protocol.

Notations: $D^2 = (\sum_{i=1}^{N} m_1(i))^2, \quad U = \sum_{i=1}^{N} m_2(i)$ (see section 3.1)

$D^2$ and $U$ are integers. By the end of CoPPSA $D^2$ and $U$ are known to all parties. Each party then calculates $Z^* = \sqrt{\frac{D^2}{U} \cdot 10^{-b}}$.

CoPPSA with incentive outputs two values $\Gamma, \Delta$ s.t. $\frac{D^2}{U} = \frac{\Gamma}{\Delta}$.

In particular, we choose the output to be:

$$(\Gamma, \Delta) = \Big(\frac{D^2}{gcd(D^2, U)}, \frac{U}{gcd(D^2, U)}\Big)$$

($gcd$ = greatest common divisor)

In CoPPSA with incentive each party gets the output $(\Gamma, \Delta)$, but not necessarily $(D^2, U)$.

### 5.3.1 Notations

- Let $p$ be a prime number.

  $\mathbb{Z}_p = \{0, ..., p-1\}$ $\mathbb{Z}_p^* = \{1, ..., p-1\}$

  $\forall x \in \mathbb{Z}_p^*$ there exists $\forall y \in \mathbb{Z}_p^*$ such that $x \cdot y = 1 \mod p$. $y$ is a **multiplicative**

**inverses** of $x$. $x^{-1} = y \mod p$

- **Extended Euclidean Algorithm** – EEA – is a deterministic polynomial-time extended version of the Euclidean algorithm. Given two integers $0 < a, 0 < b$ EEA(a,b) outputs (g,x,y) s.t.:

  1. $g = gcd(a, b)$
  2. $xa + yb = gcd(a, b)$ (x and y integers, may be negative).

  Denote: $(g, x, y) = EEA(a, b)$

- $\forall a \in \mathbb{Z}^*_p$ we can calculate $(a^{-1} \mod p)$ with EEA as follows:
  $EEA(a, p)$ outputs $(gcd(a, p), x, y)$ . Since p is prime $gcd(a, p) = 1$.

$$xa + yp = gcd(a, p) = 1 \rightarrow (xa + yp)\, mod\, p = 1 \rightarrow xa \mod p = 1$$

  Therefore: $a^{-1} = x + p \mod p$
  $(+p \mod p)$ is added because $x$ may be negative.

- Let $(n, m)$ be positive integers. $n < m$
  The **Rational Reconstruction [15] Problem**:
  Given $(n, m)$ one should find $u$ s.t. there exist integers $(r, 0 < s)$ s.t.:

$$ns = r \mod m, \quad u = \frac{r}{s}$$

  A deterministic algorithm for solving this problem in polynomial time was given by S. Wang in 1981 [21]. His solution use the Extended Euclidean Algorithm.
  This solution applies under the condition that there are 2 known constants $R$ and $S$ s.t. $|r| < R, 0 < s < S$ and $2RS < m$ . Under this condition the solution $u$ is unique.
  Notations: $(a, b) = RaRe(n, m, R, S), \quad u = \frac{a}{b} = \frac{r}{s}$

### 5.3.2 CoPPSA-V: CoPPSA With Randomization

The notations in this section follow the notations of Section 3.1.

1. All Parties consent on some value $o_{max}$.

   $o_{max}$ is the maximal umber of failures possible in any of the parties.

   $R = (10^b \cdot o_{max} \cdot N)^2$

   $S = (10^b \cdot \frac{o_{max}}{4} \cdot N)$

   The parties agree on some value $p$ s.t. $2RS < p$

2. For each party i, $i = 1, 2...., N$:

   - The party calculates: $O(i), E(i), V(i)$
   - The party round the real numbers to integers.

     $m_1(i) = \lfloor 10^b \cdot (O(i) - E(i)) \rfloor, \quad m_2(i) = \lfloor 10^b \cdot V(i) \rfloor$
   - The party samples a random number uniformly distributed): $r_i \in [0, p-1]$

3. The parties use MPC to jointly calculate the following over $\mathbb{Z}_p$:

   - $\tilde{D} = \left( (\sum_{i=1}^{N} r_i) \cdot \left( \sum_{i=1}^{N} m_1(i) \right)^2 \right) \mod p$
   - $\tilde{U} = \left( (\sum_{i=1}^{N} r_i) \cdot (\sum_{i=1}^{N} m_2(i)) \right) \mod p$
   - $(\tilde{D}, \tilde{U})$ is the common output of all parties.

4. If $\tilde{U} = 0$: Abort and start over. (happens when $(\sum_{i=1}^{N} r_i) = 0$)

5. Each party calculates $Z^*$:

   - Calculates $(\tilde{U}^{-1} \mod p)$ over $\mathbb{Z}_p^*$ using EEA
   - Calculates over $\mathbb{Z}_p$: $n = \tilde{D} \cdot \tilde{U}^{-1} \mod p$
   - Calculates $(\gamma, \delta) = RaRe(n, p, R, S), \quad u = \frac{\gamma}{\delta}$
   - $|Z^*| = \sqrt{10^{-b} u}$ over $\mathbb{R}$.

**Algorithm 2:** CoPPSA-V

**Correctness Proof**

**Claim 5.3.1.** $2RS < p \Rightarrow 2 \cdot (\sum_{i=1}^{N} m_1(i))^2 (\sum_{i=1}^{N} m_2(i)) < p$

*Proof.* $R = (10^b \cdot o_{max} \cdot N)^2, S = (10^b \cdot \frac{o_{max}}{4} \cdot N)$

From section 3.1.2 we know: $\sum_{i=1}^{N} m_2(i) < \frac{N}{4} \cdot 10^b \cdot o_{max}, \quad \sum_{i=1}^{N} m_1(i) < N \cdot o_{max} \cdot 10^b$

$$2 \cdot (\sum_{i=1}^{N} m_1(i))^2 (\sum_{i=1}^{N} m_2(i)) < 2(10^b \cdot o_{max} \cdot N)^2 (10^b \cdot \frac{o_{max}}{4} \cdot N) < 2RS < p$$

□

**Claim 5.3.2.** $n = \Big(\sum_{i=1}^{N} m_1(i)\Big)^2 (\sum_{i=1}^{N} m_2(i))^{-1} \mod p$

*Proof.* $\tilde{U} \neq 0$ so $\tilde{U}$ has multiplicative inverse in the group $\mathbb{Z}_p^*$

$$n = \tilde{D} \cdot \tilde{U}^{-1} \mod p = \Big((\sum_{i=1}^{N} r_i) \cdot \Big(\sum_{i=1}^{N} m_1(i)\Big)^2\Big)\Big((\sum_{i=1}^{N} r_i) \cdot (\sum_{i=1}^{N} m_2(i))\Big)^{-1} \mod p$$

$$= \Big(\sum_{i=1}^{N} m_1(i)\Big)^2 (\sum_{i=1}^{N} m_2(i))^{-1} \mod p$$

□

**Claim 5.3.3.** *$(\gamma, \delta) = RaRe(n, p, R, S) \rightarrow u = \frac{\gamma}{\delta}$ is correct and unique solution for the rational reconstruction problem, and $u = \frac{(\sum_{i=1}^{N} m_1(i))^2}{\sum_{i=1}^{N} m_2(i)}$ .*

*Proof.* From claim 5.3.1 we know:

$$2 \cdot (\sum_{i=1}^{N} m_1(i))^2 (\sum_{i=1}^{N} m_2(i)) < p$$

From claim 5.3.2 we know:

$$n = \Big(\sum_{i=1}^{N} m_1(i)\Big)^2 (\sum_{i=1}^{N} m_2(i))^{-1} \mod p$$

Therefore by the correction of Wang algorithm we know

$$u = \frac{\gamma}{\delta} = \frac{(\sum_{i=1}^{N} m_1(i))^2}{\sum_{i=1}^{N} m_2(i)}$$

and the value of $u$ is unique. □

**Claim 5.3.4.** $|Z^*| \approx \sqrt{10^{-b}u}$ *over* $\mathbb{R}$

*Proof.* From claim 5.3.3 we know:

$$u = \frac{(\sum_{i=1}^{N} m_1(i))^2}{\sum_{i=1}^{N} m_2(i)} = \frac{(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor)^2}{\sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor}$$

$$\sqrt{10^{-b}u} = \sqrt{\frac{(\sum_{i=1}^{N} m_1(i))^2 10^{-b}}{\sum_{i=1}^{N} m_2(i)}} = \sqrt{\frac{(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor)^2 10^{-b}}{\sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor}} \approx$$

$$\approx \sqrt{\frac{(\sum_{i=1}^{N} 10^b \cdot (O(i) - E(i)))^2 10^{-b}}{\sum_{i=1}^{N} 10^b \cdot V(i)}} = \sqrt{\frac{10^b (\sum_{i=1}^{N} O(i) - E(i))^2}{10^b \sum_{i=1}^{N} V(i)}} = \frac{|\sum_{i=1}^{N} (O(i) - E(i))|}{\sqrt{\sum_{i=1}^{N} V(i)}} = |Z^*|$$

□

Comments:

- All the secure MPC operations are in phase 3:

$$F\Big((m_{11}, m_{21}, r_1), (m_{12}, m_{22}, r_2), ..., (m_{1N}, m_{2N}, r_N)\Big) =$$

$$\Big((\sum_{i=1}^{N} r_i) \cdot \Big(\sum_{i=1}^{N} m_1(i)\Big)^2, (\sum_{i=1}^{N} r_i) \cdot (\sum_{i=1}^{N} m_2(i))\Big) \; mod\, p$$

- The common output does not include the sign (positive or negative) of $Z^*$. The parties can only calculate the absolute value $|Z^*|$.

  If $|Z^*|$ indicates a insignificant p-value (if $|Z^*| < 1.65$ then $0.05 <$ p-value), then the result has no statistical clarity and the sign is less relevant.

If $|Z^*|$ indicates a significant p-value each party can conclude the sign of $Z^*$ from it's own dataset. For sufficiently large dataset the sign of $Z_{local}$ is in coherence with the sign of $Z^*$

### 5.3.3 Does CoPPSA-V Have an Incentive for Veracity?

Notations: $D^2 = (\sum_{i=1}^{N} m_1(i))^2, \quad U = \sum_{i=1}^{N} m_2(i)$

As mentioned, by the end of CoPPSA-V each party knows two values $\Gamma, \Delta$ s.t. $(\Gamma, \Delta) = \left(\frac{D^2}{gcd(D^2,U)}, \frac{U}{gcd(D^2,U)}\right)$.

In this section we show that if $gcd(D^2, U)$ has an entropy, then CoPPSA-V has an incentive for veracity.

**Claim 5.3.5.** *CoPPSA-V has incentive for veracity.*

*Proof.* First scenario:

Consider a scenario in which party $P_l$ uses $m_{2,false}$ instead of $m_{2,true}$ as the value communicated in the MPC stage. To calculate the desired value the party should "fix" the protocol result as follows:

$$(\Gamma, \Delta)_{fixed} = \Big(\Gamma, (\Delta \cdot gcd(D^2, U) - m_{2,false} + m_{2,true})\frac{1}{gcd(D^2, U)}\Big)$$

Alternatively, $P_l$ may want to "fix" $Z^*$ directly:

$$Z^*_{fixed} = \sqrt{\frac{\Gamma}{\Big(\Delta \cdot gcd(D^2, U) - m_{2,false} + m_{2,true}\Big)\frac{1}{gcd(D^2,U)}}}$$

To calculate $(\Gamma, \Delta)_{fixed}$ or $Z^*_{fixed}$, $P_l$ should know the value of $gcd(D^2, U)$. $gcd(D^2, U)$ is a function of the random variables $m_1(1), m_2(1), m_1(2), ....$ Simply put, since $m_1(1), m_2(1), m_1(2), ...$ are random variables with a given distribution, $gcd(D^2, U)$ is also non-deterministic random variable, with distribution that is directly derived from the inputs distribution (see discussion

below).

$$\Delta_{fixed} = (\Delta \cdot gcd(D^2, U) - m_{2,false} + m_{2,true}) \frac{1}{gcd(D^2, U)}$$

$$H(\Delta_{fixed}|\Delta_{false}) = H\Big((\Delta_{false} \cdot gcd(D^2, U) - m_{2,false} + m_{2,true}) \frac{1}{gcd(D^2, U)} | \Delta_{false} = \delta\Big)$$

$$= H\Big((\delta \cdot gcd(D^2, U) - m_{2,false} + m_{2,true}) \frac{1}{gcd(D^2, U)}\Big) > 0$$

Similarly:

$$H(Z^*_{fixed}|\Gamma, \Delta) = H\Big(\sqrt{\frac{\Gamma}{\Big(\Delta \cdot gcd(D^2, U) - m_{2,false} + m_{2,true}\Big) \frac{1}{gcd(D^2, U)}}} | \Gamma = \gamma, \Delta = \delta\Big)$$

$$= H\Big(\sqrt{\frac{\gamma}{\Big(\delta \cdot gcd(D^2, U) - m_{2,false} + m_{2,true}\Big) \frac{1}{gcd(D^2, U)}}}\Big) > 0$$

Second scenario:

Consider a scenario in which $P_l$ uses $m_{1,false}$ instead if $m_{1,true}$. The "fix" should be as follows:

$$(\Gamma, \Delta)_{fixed} = \Big((\sqrt{\Gamma \cdot gcd(D^2, U)} - m_{1,false} + m_{1,true})^2 \frac{1}{gcd(D^2, U)}, \Delta\Big)$$

Like in first scenario we can show that to calculate $(\Gamma, \Delta)_{fixed}$ $P_l$ should know the value of $gcd(D^2, U)$, which is a random variable.

So we see that $Z^* = \sqrt{10^{-b\frac{\gamma}{\delta}}}$ is a function with incentive for veracity and that CoPPSA-V has an incentive for veracity. □

**Discussion**

Could $gcd(D^2, U)$ have no entropy?

To show that $gcd(D^2, U)$ must have an entropy, we will show that:

$$Pr(gcd(D^2, U) = 1) > 0$$

$$Pr(gcd(D^2, U) \neq 1) > 0$$

In this case $(X, Y) = (\frac{D^2}{gcd(D^2,U)}, \frac{U}{gcd(D^2,U)}) = (D^2, U)$. To estimate the probability that $gcd(D^2, U) = 1$ consider Euler observation regarding Coprimality[16]:

**Theorem 5.3.6.** *Let $N$ be a positive integer. $a, b$ are random variables with uniform distribution in the range $\{1, ..., N\}$. Let $P_N$ be the probability that $a$ and $b$ are coprime integers. Observation:*

$$\lim_{N\to\infty} P_N = \frac{6}{\pi^2} \approx 0.607$$

Simply put, the probability of two integers uniformly sampled from sufficiently large range to be coprime integers is approximately $\frac{6}{\pi^2} = 0.607$.

The two integers $(\sum_{i=1}^{N} \lfloor 10^b \cdot (O(i) - E(i)) \rfloor)^2, \sum_{i=1}^{N} \lfloor 10^b \cdot V(i) \rfloor$ are large enough and close to uniform in some arbitrary range. We can rely on this approximation and assume that $Pr(gcd(D^2, U) = 1) > 0$ and $Pr(gcd(D^2, U) \neq 1) > 0$ . Therefore $gcd(D^2, U)$ is not a constant.

# Part II

# Statistical Stability For Logrank Test

# Chapter 6

# Logrank Stability

When applying the log-rank test to a set of data, we are implicitly assuming that the labels of the subjects are correct. In reality, however, this assumption is often compromised. In some cases the label assignment is, indeed, rather straight forward. This is typically the case in the assignment to treatment arms and when actual compliance is verifiable. In other cases it may be much less well defined.

This is the case when label assignment is determined by a human judgment, for example based on inspection by pathologists, which is often prone to errors.

As a second example consider labeling that follows a machine decision. Three recent studies [10, 9, 7] introduce machine learning models to determine breast cancer sub-types. They all reported around 70% accuracy.

In the context of survival analysis, wrong sample labels can lead to dramatically different statistical assessments. Consider the MAINZ cohort, [19], that describes survival data for breast cancer patients. As can be seen in Figure 6.1, Luminal A patients have better prognosis than the other types. We note a significant difference in the Luminal A prognosis with p-value = 0.014. Now, what will the effect be, on the resulting p-value, of changing

one Luminal A label out of the 200 samples (0.5%) in the MAINZ cohort. Figure 6.1 shows the Kaplan-Meier graphs, before (left) and after (right) the change, and the corresponding p-values. While the plots themselves seemed very similar, the p-value changed from 0.014 to 0.029.

In this work we address the uncertainty that arises from general labeling errors and label instability. Given survival data with $n$ samples and an error rate, $\alpha$, we find the minimum and maximum log-rank p-values that can result from changing the labels of at most $\alpha n$ samples. These minimum and maximum p-values, $P_L$ and $P_U$, define a stability interval $[P_L,P_U]$. To make our analysis less sensitive to extreme cases we support the use of a confidence level, $1-\delta$, to further narrow our interval. The main contributions of our work are as follows:

- A definition of labeling errors stability intervals for statistical tests.
- A procedure that, given data and a bound on the labeling error, calculates a stability interval for the log-rank test.

*Bioinformatics*, 37(23), 2021, 4451–4459
doi: 10.1093/bioinformatics/btab495
Advance Access Publication Date: 13 July 2021
Original Paper

Genetics and population analysis

# On the stability of log-rank test under labeling errors

**Ben Galili [1,*], Anat Samohi[2,*] and Zohar Yakhini[1,2,*]**

[1]Faculty of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel and [2]Arazi School of Computer Science, Interdisciplinary Center, Herzliya, Israel

*To whom correspondence should be addressed.
Associate Editor: Russell Schwartz

Received on January 21, 2021; revised on June 25, 2021; editorial decision on June 30, 2021; accepted on July 2, 2021

## Abstract

**Motivation:** Log-rank test is a widely used test that serves to assess the statistical significance of observed differences in survival, when comparing two or more groups. The log-rank test is based on several assumptions that support the validity of the calculations. It is naturally assumed, implicitly, that no errors occur in the labeling of the samples. That is, the mapping between samples and groups is perfectly correct. In this work, we investigate how test results may be affected when considering some errors in the original labeling.

**Results:** We introduce and define the uncertainty that arises from labeling errors in log-rank test. In order to deal with this uncertainty, we develop a novel algorithm for efficiently calculating a stability interval around the original log-rank *P*-value and prove its correctness. We demonstrate our algorithm on several datasets.

**Availability and implementation:** We provide a Python implementation, called LoRSI, for calculating the stability interval using our algorithm https://github.com/YakhiniGroup/LoRSI.

**Contact:** benga9@gmail.com or anatsamohi@gmail.com or zohar.yakhini@gmail.com

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

The comparison of different treatments or, more generally, policies or protocols, in terms of survival rates or in terms of success rates is a central aspect of investigating these regimes and of taking related decisions. There are two approaches that are generally taken in analyzing survival data. The first uses a permutational null distribution (Heimann and Neuhaus, 1998; Vandin *et al.*, 2015) and is more appropriate for imbalanced data. The second, more popular approach, uses a conditional null model, based on the hypergeometric distribution. This second approach is also the focus of this article. The log-rank test was introduced by Mantel (1966) and is extensively used since then. It is a standard tool in survival analysis, e.g. Kleinbaum and Klein (2012). In Tourneau *et al.* (2015), reporting on the SHIVA study, the log-rank test was used to determine whether the use of several targeted therapies outside their intended indications will improve progression-free survival in cancer. In Pitt *et al.* (1999), the authors used log-rank test to conclude that the use of Spironolactone is effective to lower the risk of death in patients who suffered from severe heart failure. Galili *et al.* (2021) investigate efficient gene signatures that characterize a breast cancer subtype related to the patient's immune response. The signature is optimized using a survival criterion based on the log-rank test. Levy-Jurgenson *et al.* (2020) report how cancer intratumor heterogeneity can affect patient survival.

When applying the log-rank test to a set of data, we are implicitly assuming that the association of a subject, or, more generally, a sample, to one of the two labels, is not in doubt. In reality, however, this assumption is often compromised. In some cases, the label assignment is, indeed, rather straight forward. This is typically the case in the assignment to treatment arms. In other situations, it may be much less well defined.

This is the case, as a first example, when label assignment is determined by a human judgment, e.g. based on inspection by pathologists, which is often prone to errors. Literature explicitly reports inconsistency in pathology. Jackson *et al.* (2017) report a study that found that the decision of the same pathologist varied when examining the same samples in different times. They showed that two diagnostic calls of the same pathologist, separated by at least 9 months, on the same biopsy, have an agreement rate of 92% (95% CI 88–95%) for invasive breast cancer and even less for other breast cancer types. Jackson *et al.* (2017) also showed that for different pathologists testing the same biopsy the agreement rates dropped by additional 3–10%. Elmore *et al.* (2017) reported similar results. In a different context, any subjective scoring approach, such as the Eastern Cooperative Oncology Group (ECOG) score, as used in Loprinzi *et al.* (1994), depends, based on the same principles, on the individual making the calls.

As a second example consider labeling that follows a machine decision. Three recent studies (Ha *et al.*, 2019; Islam *et al.*, 2020; Jaber *et al.*, 2020) introduce machine-learning models to determine breast cancer subtypes. They all reported around 70% accuracy.

Finally, consider sample labeling, which is based on the results of some molecular measurement assay. Ebbert *et al.* (2011) showed how intrinsic errors in the laboratory process, specifically in gene

expression profiling, affect the final results. They test this on PAM50 results and reported around 5% error in the classification.

In the context of survival analysis, wrong sample labels can lead to dramatically different statistical assessments. Consider the MAINZ cohort, Schmidt *et al.* (2008), that describes survival data for breast cancer patients. As extensively reported, including in Fallahpour *et al.* (2017) and Howlader *et al.* (2018), Luminal A patients have better prognosis than the other types. This can be seen also in the MAINZ cohort, see the left panel in Figure 1. We note a significant difference in the Luminal A prognosis with *P*-value =0.014. Now, what will the effect be, on the resulting *P*-value, of changing one Luminal A label out of the 200 samples (0.5%) in the MAINZ cohort? Figure 2 shows the original data and the data after one label change. Figure 1 shows the Kaplan–Meier graphs, before (left) and after (right) the change, and the corresponding *P*-values. Examining Figure 1 shows a dramatic change in the *P*-value from 0.014 to 0.029, when it is not so simple to notice any change in the plots themselves. Expanding this observation to the actual labeling error, e.g. as reported in Ebbert *et al.* (2011) for breast cancer subtypes, can lead to even more dramatically changes in the *P*-value.

Previous investigations addressed several aspects of uncertainty in survival analysis. Heterogeneity between individuals is not taken into account in the basic form of log-rank test. To address this bias, Hougaard (1995) introduced the concept of frailty models for survival analysis. Under this approach, the null model does not assume that the distribution of time to event is the same for all subjects. In order to overcome the unobserved heterogeneity in the survival data the frailty models use random effect to create different time to event distributions.

Addressing a different issue, it is common to report (and plot) confidence intervals for each of the observed hazard ratios, resulting in a confidence envelope around the survival lines. In Vandin *et al.* (2015), the authors demonstrated that asymptotic approximation, as in log-rank test, can be misleading when the two groups under consideration have very different sizes. They introduced a novel approach to accurately calculate the log-rank *P*-value regardless of the group sizes. Splitting subjects to two groups, in order to determine an association between the split and, potentially, low risk and high risk, is an important task in the context of survival analysis. Standard studies use treatment types, protocols etc. When studying a quantitative potential determinant of survival, we are often interested in splitting according to that quantity. For example, the expression level of some gene or maybe BMI. Trying all possible cut-points (thresholds) is not practical due to multiple testing problems. In an important paper treating this issue, Hothorn and Lausen (2003) developed a method for calculating an upper bound on the log-rank test *P*-value, which efficiently takes into account multiple testing. This approach checks different label assignments, similar to our work, but limits the splits being considered. In addition, it focuses on finding the best split. In this work, we address general labeling and not necessarily such which is driven by a quantitative feature. Our study also considers completely general labeling changes and not ones related to consistent threshold splits.

In this work, we address, for the first time, the uncertainty that arises from general labeling errors and label instability. Given survival data with $n$ samples and an error rate, $\alpha$, we find the minimum and maximum log-rank *P*-values that can result from changing the labels of at most $\alpha n$ samples. These minimum and maximum

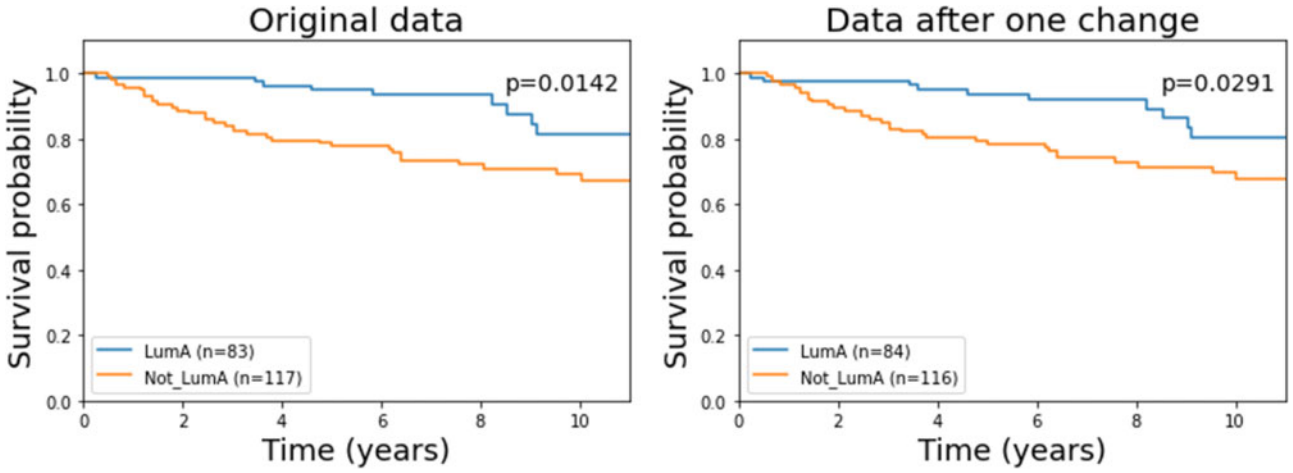


**Fig. 1.** Kaplan–Meier curve and log-rank *P*-value on the MAINZ cohort—Luminal A versus not Luminal A. On the left the original data and on the right the data after one change

Original data

| Time (days) | 30 | 30 | 90 | 180 | ... | 6030 | 6090 | 6150 | 7200 |
|---|---|---|---|---|---|---|---|---|---|
| Event | 0 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 |
| Group | LumA | Not | LumA | Not | ... | Not | Not | Not | Not |

Data after one change

| Time (days) | 30 | 30 | 90 | 180 | ... | 6030 | 6090 | 6150 | 7200 |
|---|---|---|---|---|---|---|---|---|---|
| Event | 0 | 0 | 1 | 1 | ... | 0 | 0 | 0 | 0 |
| Group | LumA | Not | LumA | LumA | ... | Not | Not | Not | Not |

**Fig. 2.** MAINZ cohort data, before and after one change in the labels

*P*-values, $P_L$ and $P_U$, define a stability interval $[P_L, P_U]$. To make our analysis less sensitive to extreme cases, we support the use of a confidence level, $1-\delta$, to further narrow our interval. The main contributions of our work are as follows:

- A definition of labeling errors stability intervals for statistical tests.
- A procedure that, given data and a bound on the labeling error, calculates a stability interval for the log-rank test.
- A software implementation of the above procedure.
- https://github.com/YakhiniGroup/LoRSI.
- Applications to several example cases.

## 2 Materials and methods

### 2.1 The statistical framework for log-rank stability

#### 2.1.1 Preliminaries

We first set the notation for the log-rank test, in the context of the conditional null distribution (Mantel, 1966), as will be used in the rest of the manuscript.

- Consider time and event data $D$ (survival data) with a partition labeling $\lambda_0$ [a binary vector mapping subjects into the groups $A$ (0) & $B$ (1)] over $n$ subjects.
- Let $j = 1, \ldots, J$ be the distinct times of observed events in either group.
- Let $n_{A,j}$, $n_{B,j}$ be the number of subjects 'at risk' (who have not yet had an event nor have been censored) at the time of occurrence of the $j$th event in the two groups, respectively.
- Let $O_{A,j}$, $O_{B,j}$ be the random variables representing the observed number of events in each group at time $j$.
- Denote $n_j = n_{A,j} + n_{B,j}$, the number of at-risk subjects at time $j$.
- Denote $o_j = O_{A,j} + O_{B,j}$, the number of actual events observed at time $j$.
- Let T be the time of failure of a subject. $P(T = t)$ is the probability distribution function of $T$. The survival function is defined as $S(t) = 1 - P(T < t) = 1 - F(t)$.
- In log-rank testing, we are working under the null model that assumes that the two groups have identical survival functions, $S_A(t) \equiv S_B(t)$
- We then have

$$O_{A,j} \sim HG(n_j, n_{A,j}, o_j).$$

  Similar for group $B$ (HG stands for Hypergeometric).
- The null model also assumes that the variables $O_{A,j}$ are (collectively) independent.
- The expected value and the variance of $O_{A,j}$ under the null model are:

$$E_{A,j} = \frac{n_{A,j}}{n_j} o_j$$
$$V_{A,j} = \frac{n_{A,j}}{n_j} o_j \left(\frac{n_j - o_j}{n_j}\right)\left(\frac{n_j - n_{A,j}}{n_j - 1}\right).$$

  Similar for group B.
- Putting everything together, for all $j = 1, \ldots, J$, the log-rank statistic compares $O_{A,j}$ to their expected values $E_{A,j}$ under the null model. The statistic is defined as:

$$Z_A = \frac{O - E}{\sqrt{V}},$$

where:

$$O = \sum_{j=1}^{J} O_{A,j} \quad E = \sum_{j=1}^{J} E_{A,j} \quad V = \sum_{j=1}^{J} V_{A,j}.$$

Similar for group B.

If $J$ is sufficiently large and the partition into $A$ and $B$ is reasonably balanced (see e.g. Vandin *et al.*, 2015) then, $Z$ is approximately distributed as $N(0, 1)$. This allows us to compute a $P$-value for the comparative survival data $D$, using the value actually observed for $O$, which we denote $o = o(D)$. This $P$-value is denoted by $LR(D, \lambda_0)$. By extension $LR(D, \lambda)$ will denote the log-rank $P$-value that would be obtained for any different partition labeling $\lambda$.

#### 2.1.2 Definition of the log-rank stability interval

We now define a log-rank stability interval for given survival data and two parameters $\alpha > 0$ and $\delta \geq 0$.

- Again, consider time and event data $D$ with a partition labeling $\lambda_0$ (mapping subjects into the groups A & B) over $n$ subjects. Recall that $LR(D, \lambda_0)$ is the log-rank $P$-value computed for this data.
- Let $0 < \alpha < 1$. Given a different binary labeling $\lambda$, we say that $\lambda$ is an $\alpha$-modification of $\lambda_0$ if the labels have changed in less than a fraction $\alpha$ of the samples.
- Formally, $H(\lambda, \lambda_0) \leq \alpha \cdot n$, where $H$ is the Hamming distance.
- Let $B(\lambda_0, \alpha)$ be the set of all possible labeling partitions $\lambda$ that are $\alpha$-modifications of $\lambda_0$.
- Let $0 \leq \delta < 1$. We want to compute a tight interval $[p_L, p_U]$ in the following sense: $p_U$ should be the smallest number for which $LR(D, \lambda) \in [p_L, p_U]$ holds for a $1 - \delta$ fraction of $\lambda \in B(\lambda_0, \alpha)$. Note that under this definition, we require tightness on the right hand side, which is, in practice, taking a conservative approach, the more interesting case (see Section 4).
- The interval defined above is the stability interval for the two parameters $\alpha > 0$ and $\delta > 0$ and the input data. We write:

$$SI(D, \lambda_0, \alpha, \delta) = [p_L, p_U]. \quad (1)$$

For example, given data $D$ with $n = 100$ (100 samples), $\delta = 0.05$ and $\alpha = 0.01$, we want to compute an interval SI so that for 95% of the single label changes (1% change) the log-rank $P$-value will fall in SI.

### 2.2 Computing stability intervals for log-rank test

In this section, we describe our algorithmic approach and prove its correctness.

#### 2.2.1 Algorithm: LoRSI

We start by some definitions and notations.

Let:

- $D$– the dataset. Consisting of three vectors of length $n$:
  a. $e$: event/censored descriptor. Indicates whether event or censored occurred.
  b. $t$: time. The time from the beginning of an observation period to an event, censored or end of the study.
  c. $l$: group. Indicates the group of the subject.
- $d = (l(d), t(d), e(d))$ represents a single instance: an instance $d \in D$ is defined by three quantities—the group label $l(d)$, the time $t(d)$ and the event/censored descriptor $e(d)$.
- $F$ : the set of interest (the Focus set).
- $B$ : the other set (the Background set).
- The set $F$ is typically the one that has a better survival rate. That is: $Z_F < Z_B$. In this article, we also take this approach and

therefore $F$ is the set that has the better survival rate in the input data, $D$.

We further define the following subsets of $F$ and $B$:

- $EF$: the events of the group $F$, ordered from the earliest to the latest.
- $CF$: the censored samples of the group $F$, ordered from the earliest to the latest.
- $EB$: the events of the group $B$, ordered from the earliest to the latest.
- $CB$: the censored samples of the group $B$, ordered from the earliest to the latest.

Note that $F = EF \cup CF$ and $B = EB \cup CB$.

Now define the prefixes and sufixes of these ordered subsets as follows:

- $EF_L(i) =$ the samples $EF(1), \ldots, EF(i)$
- $EB_L(i) =$ the samples $EB(|EB| - i + 1), \ldots, EB(|EB|)$
- $CB_L(i) =$ the samples $CB(|CB| - i + 1), \ldots, CB(|CB|)$
- $EF_U(i) =$ the samples $EF(|EF| - i + 1), \ldots, EF(|EF|)$
- $EB_U(i) =$ the samples $EB(1), \ldots, EB(i)$
- $CF_U(i) =$ the samples $CF(|CF| - i + 1), \ldots, CF(|CF|)$.

Following standard notation for the set of types of denominator $k$ over a three letter alphabet (Cover, 1999), we denote:

$$T(k,3) = \{(i_1, i_2, i_3) : i_1 + i_2 + i_3 = k\}.$$

Note that:

$$|T(k,3)| = \binom{k+2}{2}.$$

**Definition 1.** The set of $P_U$ candidates is defined by:

$$C_U = \{(EF_U(i_1) \cup EB_U(i_2) \cup CF_U(i_3)) : (i_1, i_2, i_3) \in T(k,3)\}.$$

We will show that this is the collection of candidate sample sets of size $k$, amongst which we will identify the set of samples that, if swapped, will lead to the most extreme positive change in the $P$-value. Note that the size of $C_U$ is the same as that of $T(k,3)$, namely $\binom{k+2}{2}$.

Similarly:

**Definition 2.** The set of $P_L$ candidates is defined by:

$$C_L = \{(EF_L(i_1) \cup EB_L(i_2) \cup CB_L(i_3)) : (i_1, i_2, i_3) \in T(k,3)\}.$$

Algorithm 1 describes the Log-Rank Stability Interval Algorithm (LoRSI) for finding $P_L$ and $P_U$, where $\alpha = \frac{k}{n}$. For $P_U$, the idea of the algorithm is to iterate over the set of all relevant sets of $k$ changes. The size of this collection is relatively small, namely $\binom{k+2}{2}$, due to the monotonicity effect on the $z$-score in each one of the groups $E_F, C_F$ & $E_B$ as proven below in Section 2.2.2. In each iteration, our procedure calculates the $P$-value after changing the labels of the current $k$ subjects. Finally, it selects the max $P$-value among the $\binom{k+2}{2}$ candidates. Note that, if we consider only one label change ($k = 1$), then, the SI (both sides) is determined by only six candidates, three for $P_U$ and three for $P_L$ (see Fig. 3). In the Supplementary Material, we describe the LoRSI algorithm, where $\alpha = \frac{1}{n}$ and for any $\delta > 0$.

**Algorithm 1:** LoRSI pseoducode.

**Log-Rank Stability Interval**
**input:** Dataset—$D$, $\alpha = \frac{k}{n}$
**output:** Stability Interval $[p_L, p_U]$
$p_L$*_candidates* $= \varnothing$
$p_U$*_candidates* $= \varnothing$
//each of these sets will hold all $\binom{k+2}{2}$ relevant $P$-values
**for** *current_set_of_changes in* $C_U$ **do**
  //see Definition 1 for $C_U$
  $p =$ log-rank $P$-value after swapping the labels of all the $k$ samples in *current_set_of_changes*
  $p_U$*_candidates*.append ($p$)
**end**
**for** *current_set_of_changes in* $C_L$ **do**
  //see Definition 2 for $C_L$
  $p =$ log-rank $P$-value after swapping the labels of all the $k$ samples in *current_set_of_changes*
  $p_L$*_candidates*.append ($p$)
**end**
$p_U = \max(p_U\text{\_candidates})$
$p_L = \min(p_L\text{\_candidates})$
**return** $p_L$, $p_U$

### 2.2.2 Correctness

In this section, we prove the correctness of Algorithm 1. This, in essence, is the content of Theorem 1, stated at the end of this section. We start with some definitions and notations.

- Let $z_0$ be the original $Z$-statistic obtained from the input labeling.
- Now consider a labeling swap for the instance $d$. That is, if in $\lambda_0$, the instance $d$ is in the group $F$, then, it is swapped to $B$ and symmetrically otherwise. This swap will affect the value of $Z$ calculated for the new data. Let

$$z_{new}(D,d) = \frac{o_{new} - E_{new}}{\sqrt{V_{new}}},$$

  where $o_{\text{new}}$, $E_{\text{new}}$ and $V_{\text{new}}$ are obtained for the swapped data as described in the preliminaries.
- We are specifically interested in the resulting change in the observed value of $Z$, which we denote

$$\Delta z(D,d) = z_{new}(D,d) - z_0.$$

Let $(Y_1, \ldots, Y_n)$ be a set of RVs. We say that $(Y_1, \ldots, Y_n)$ is an independent hypergeometric set (IHS) if:

1. $Y_1, \ldots, Y_n$ are (collectively) independent.
2. $\forall i\ Y_i \sim HG(N_i, B_i, n_i)$.
3. $\forall i\ Var(Y_i) > 0$.

For a single RV $X$ let

$$Z(X) = \frac{X - E(X)}{\sqrt{V(X)}}$$

and then, for a set of independent RVs,

$$Z(X_1, \ldots, X_n) = Z(\sum_{i=1}^{n} X_i).$$

We note that if $(Y_1, \ldots, Y_n)$ is an IHS and if $n$ is sufficiently large then the distribution of $Z(Y_1, \ldots, Y_n)$ is approximately standard

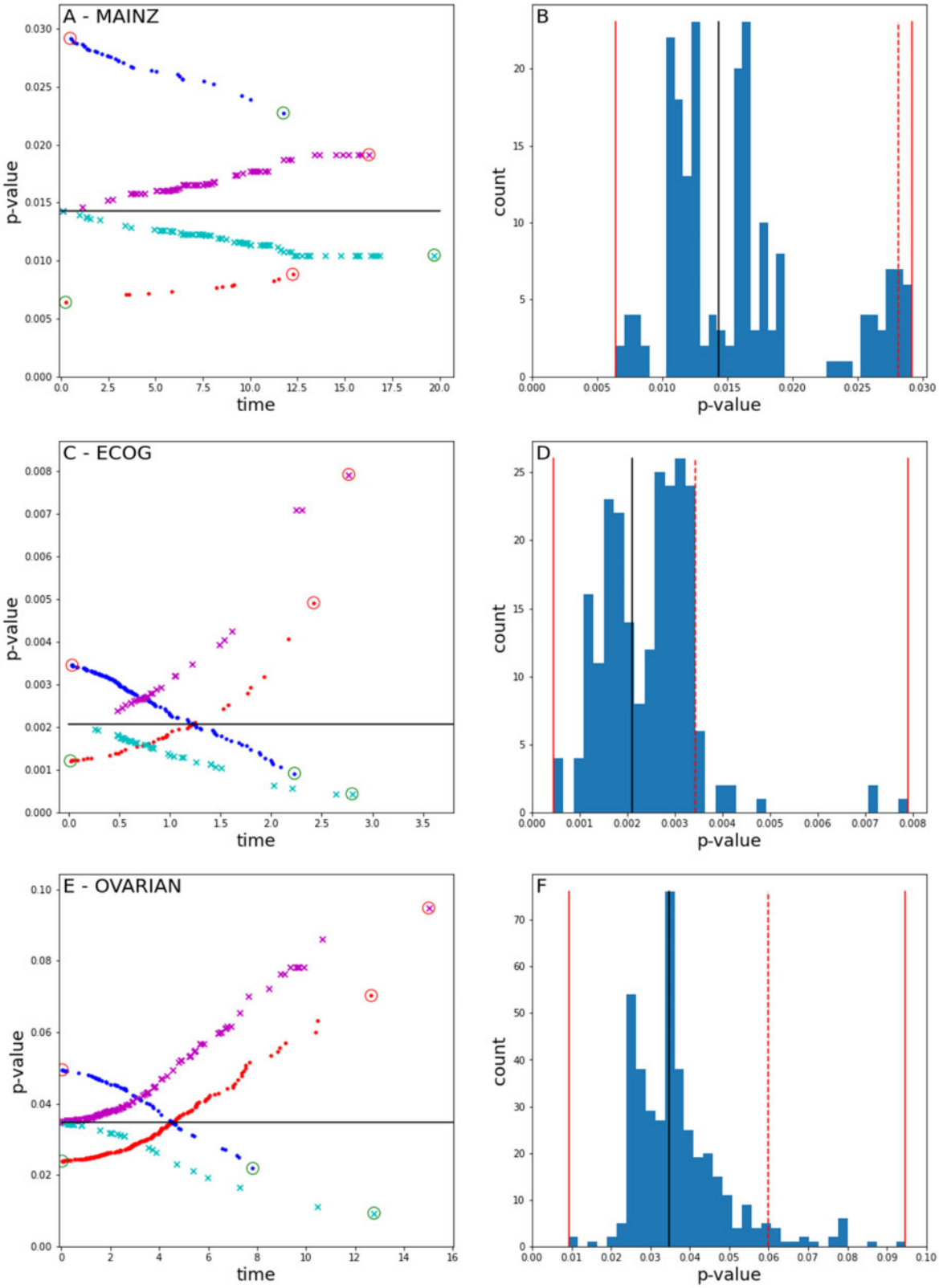


**Fig. 3.** Log-rank stability analysis for single label changes. The figure represents results for three datasets, as described in the text. Each of the depicted datasets consists of two groups (**F**)—the (actual, as per the original data) group of patients with good prognosis, and (**B**)—the bad prognosis group. Each data point is either an event or a censored point. The combination of the group and the event type leads to four categories of patients. The scatter plots provide a visual representation of the effect, on the *P*-value, that follows from changing the label of a single sample (i.e. $\alpha = \frac{1}{n}$). We can observe the monotonicity of the effect, with a direction depending on the category, as proven in Section 2. For each dataset, we indicate the original (non-swap) *P*-value (solid black lines on the right panel), the numbers $P_L$ and $P_U$ (solid red lines) for $\delta$=0 and the number $P_U$ for $\delta = 0.05$ (dashed red lines). Sample categories, in the scatter plots are represented by shape and color: blue dots—event swap from (B) to (F), red dots—event swap from (F) to (B), cyan Xs—censored sample swap from (B) to (F) and purple Xs—censored sample swap from (F) to (B). The green and red circles represent $P_L$ and $P_U$ candidates, respectively, at $\delta = 0$

normal (Lindeberg, 1922). We also note that, as stated above, the variables $O_{A,j}$, where $1 \leq j \leq J$, constitute an IHS.

For an observation $x$, derived from an RV $X$, we further define the Z-transformed value:

$$z(x) = \frac{x - E(X)}{\sqrt{V(X)}}.$$

For a set observation we now define

$$z(X_1 = x_1, \ldots, X_n = x_n) = z(\sum_{i=1}^{n} x_i).$$

For a random variable $X$ and a number $x \in \mathbb{R}$, we use the notation $CDF(X, x)$ to represent the cumulative distribution of $X$ at $x$. Or, in other words: $CDF(X, x) = P(X \leq x)$.

Claim 1. *Given two RVs X, Y where:*

$$X \sim HG(N, B, n), Y \sim HG(N, B, m)$$

*and*

$$n < m$$

*then*

$$\forall b \leq n,\ CDF(Y, b) < CDF(X, b).$$

See Supplementary Material for a proof.

Claim 2. *Consider two RVs X, Y where:*

$$X \sim HG(N, B, n), Y \sim HG(N, B, m).$$

*Let $T_1, \ldots, T_k$ be k RVs where both $(X, T_1, \ldots, T_k)$ and $(Y, T_1, \ldots, T_k)$ are IHS. Denote $\mu_i = E(T_i)$ and $\sigma_i = \sqrt{V(T_i)}$.*

*Let:*

$$Z_1 = Z(Y, T_1, \ldots, T_k), z_1 = z(Y = a, T_1 = t_1, \ldots, T_k = t_k)$$
$$Z_2 = Z(X, T_1, \ldots, T_k), z_2 = z(X = a, T_1 = t_1, \ldots, T_k = t_k)$$

*If*

$$n < m$$

*then:*

1. $CDF(Z_1, z_1) < CDF(Z_2, z_2)$
2. *For sufficiently large values of k (*which *is the interesting case, in the context of log-rank, see comment after the proof), we also have:*

$$z_1 < z_2.$$

See Supplementary Material for a proof.

As noted above, we are interested in working with large values of $k$ in the context of log-rank. Without this assumption, the second part of Claim 2 is not necessarily true. For example, for $k = 1$, consider:

- $T_1 \sim HG(90, 2, 1)$ with observed value $t_1 = 1$
- $X \sim HG(100, 1, 50)$ with observed value $a = 1$
- $Y \sim HG(100, 1, 99)$ with observed value $a = 1$,

which yields: $z_1 = 5.554$, $z_2 = 2.835$.

Claim 3. *Let $d_{j_1}$ and $d_{j_2}$ be censored samples in D from group F. Let $z_1 = z_{new}(D, d_{j1})$ and $z_2 = z_{new}(D, d_{j2})$. If $time(d_{j_1}) < time(d_{j_2})$ then $z_0 < z_1 < z_2$, and therefore:*

$$0 < \Delta z(D, d_{j1}) < \Delta z(D, d_{j2}).$$

*Similarly, if the censored samples, $d_{j_1}$ and $d_{j_2}$, come from group B then if $time(d_{j_1}) < time(d_{j_2})$ then $z_0 > z_1 > z_2$, and therefore:*

$$0 > \Delta z(D, d_{j1}) > \Delta z(D, d_{j2}).$$

**Proof.** We use the fact that the random variables $O_{F,j}$ as defined in the log-rank setup constitute an IHS. Swapping $d_{j1}$ from $F$ to $B$ leads to a change in the at risk numbers $n_{F,j}\ \forall j \leq j_1$. More specifically each one of them is decreased by 1. Nothing changes for the later indices. Assuming that $J$ is sufficiently large, we now iteratively use Claim 2. In every iteration, we decrease $n_{F,j}$ by 1, starting at $j = 1$ and ending at $j = j1$. At every index $j$ let $O_{F,j}$ and $\tilde{O}_{F,j}$ be the hypergeometric variables representing the number of events at time $j$ before and after a hypothetical swap at $j$, respectively. Claim 2 therefore applies, at every iteration $j$, with $O_{F,j}$ and $\tilde{O}_{F,j}$ playing the role of $Y$ and $X$, respectively, and $\tilde{O}_{F,i}$ with $1 \leq i \leq j-1$ and $O_{F,i}$ with $j+1 \leq i \leq J$ playing the role of the $T_i$s. We, thus, get $z_0 < z_1$.

Similarly, since $j_1 < j_2$ the swap of $d_{j_2}$ will affect all at risk numbers above as well as several others $n_{F,j}\ s.t\ j_1 < j \leq j_2$. Continuing the above iterations, we therefore have $z_1 < z_2$.

When swapping away from group $B$, the effect of the swap will be to increase the at risk numbers, leading to the reverse inequalities. ■

Claim 4. *Consider the RVs $X_1$, $X_2$ and $Y_1$, $Y_2$ where:*

$$X_1 \sim HG(N, B, l), Y_1 \sim HG(N, B, l)$$
$$X_2 \sim HG(M, C, n), Y_2 \sim HG(M, C, m).$$

*Let $T_1, \ldots, T_{k-1}$ be k −1 RVs where both $(X_1, X_2, T_1, \ldots, T_{k-1})$ and $(Y_1, Y_2, T_1, \ldots, T_{k-1})$ are IHS. Denote $\mu_i = E(T_i)$ and $\sigma_i = \sqrt{V(T_i)}$.*

*Let:*

$$z_1 = z(Y_1 = a_1 - 1, Y_2 = a_2, T_1 = t_1, \ldots, T_{k-1} = t_{k-1})$$
$$z_2 = z(X_1 = a_1, X_2 = a_2 - 1, T_1 = t_1, \ldots, T_{k-1} = t_{k-1}).$$

*If*

$$n < m$$

*then*

$$z_1 < z_2$$

**Proof.** First, we write explicitly:

$$Z_1 = \frac{Y_1 - \mu_{Y_1} + Y_2 - \mu_{Y_2} + T_1 - \mu_1 + \cdots + T_{k-1} - \mu_{k-1}}{\sqrt{\sigma_{Y_1}^2 + \sigma_{Y_2}^2 + \sigma_1^2 + \cdots + \sigma_{k-1}^2}}$$
$$Z_2 = \frac{X_1 - \mu_{X_1} + X_2 - \mu_{X_2} + T_1 - \mu_1 + \cdots + T_{k-1} - \mu_{k-1}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2 + \sigma_1^2 + \cdots + \sigma_{k-1}^2}}$$
$$z_1 = \frac{a_1 - 1 - \mu_{Y_1} + a_2 - \mu_{Y_2} + t_1 - \mu_1 + \cdots + t_{k-1} - \mu_{k-1}}{\sqrt{\sigma_{Y_1}^2 + \sigma_{Y_2}^2 + \sigma_1^2 + \cdots + \sigma_{k-1}^2}}$$
$$z_2 = \frac{a_1 - \mu_{X_1} + a_2 - 1 - \mu_{X_2} + t_1 - \mu_1 + \cdots + t_{k-1} - \mu_{k-1}}{\sqrt{\sigma_{X_1}^2 + \sigma_{X_2}^2 + \sigma_1^2 + \cdots + \sigma_{k-1}^2}}.$$

By definition $\mu_{Y_1} = \mu_{X_1}$ and $\sigma_{Y_1} = \sigma_{X_1}$. Let $b_1 = a_1 - 1$. We rearrange the term to get:

$$z_1 = \frac{b_1 - \mu_{Y_1} + a_2 - \mu_{Y_2} + t_1 - \mu_1 + \cdots + t_{k-1} - \mu_{k-1}}{\sqrt{\sigma_{Y_1}^2 + \sigma_{Y_2}^2 + \sigma_1^2 + \cdots + \sigma_{k-1}^2}}$$
$$z_2 = \frac{b_1 - \mu_{Y_1} + a_2 - \mu_{X_2} + t_1 - \mu_1 + \cdots + t_{k-1} - \mu_{k-1}}{\sqrt{\sigma_{Y_1}^2 + \sigma_{X_2}^2 + \sigma_1^2 + \cdots + \sigma_{k-1}^2}}.$$

We now apply Claim 2, with: $b_1 = t_k$, $\mu_{Y_1} = \mu_k$, $\sigma_{Y_1} = \sigma_k$, $a_2 = a$, $\mu_{Y_2} = \mu_Y$, $\sigma_{Y_1} = \sigma_Y$, $\mu_{X_2} = \mu_X$, $\sigma_{X_2} = \sigma_X$ and get $z_1 < z_2$. ■

Claim 5. *Let $d_{j_1}$ and $d_{j_2}$ be events in D from group F.*
*Let $z_1 = z_{new}(D, d_{j1})$ and $z_2 = z_{new}(D, d_{j2})$.*
*If $time(d_{j_1}) < time(d_{j_2})$ then $z_1 < z_2$, and therefore:*

$$\Delta z(D, d_{j1}) < \Delta z(D, d_{j2}).$$

*Similarly, if the events, $d_{j_1}$ and $d_{j_2}$, come from group B then if $time(d_{j_1}) < time(d_{j_2})$ then $z_1 > z_2$, and therefore:*

$$\Delta z(D, d_{j1}) > \Delta z(D, d_{j2}).$$

**Proof.** We once again use the fact that the random variables $O_{F,j}$ as defined in the log-rank setup constitute an IHS. Swapping $d_{j1}$ from $F$ to $B$ leads to a change in the at risk numbers $n_{F,j}\, \forall j \leq j_1$. More specifically each one of them is decreased by 1. Similarly, since $time(d_{j_1}) < time(d_{j_2})$ the swap of $d_{j_2}$ will affect all at risk numbers above as well as several others, namely $n_{F,j}\, s.t\, j_1 < j \leq j_2$. In addition, $o_{F,j_1}$ becomes $o_{F,j_1} - 1$ when swapping $d_{j1}$ and $o_{F,j_2}$ becomes $o_{F,j_2} - 1$ when swapping $d_{j2}$. Therefore, $o_{Fnew} = o_F - 1$ in both swaps.

Now, let $Y_1$ and $Y_2$ be $O_{F,j_1}$ and $O_{F,j_2}$ after swapping $d_{j1}$ from $F$ to $B$, respectively. In addition, let $X_1$ and $X_2$ be $O_{F,j_1}$ and $O_{F,j_2}$ after swapping $d_{j2}$ from $F$ to $B$, respectively. By iteratively using Claim 4 and assuming that $J$ is sufficiently large, we get $z_1 < z_2$.

In the case of swapping away from group $B$, the effect of the swap will be to increase both the at risk numbers and the observed $o_{Fnew}$, and therefore we get the reverse inequalities. ■

In summary, we proved that changing one sample will lead to a monotonic effect on the z-score and therefore on the log-rank $P$-value.

Now consider the case of $k$ swaps. We claim that the $k$ instances that yield the most extreme positive change in the $P$-value is one of the candidates in $C_U$. Let $\lambda^*$ be the labeling that, indeed, yields the largest $LR(D, \lambda)$ within $B(\lambda_0, \frac{k}{n})$. To see why the above claim holds assume, WLOG, that $\lambda^*$ swaps some instance $EB_U(j)$ but does not swap $EB_U(i)$ for some $i<j$. By the monotonicity proven above (Claim 5), we can swap $EB_U(i)$ instead of $EB_U(j)$ and get a larger $\Delta z$. A similar argument holds for an assumed usage, by $\lambda^*$, of a non-continuous suffix of EF and CF, respectively. Furthermore, a similar argument holds for the left side of the interval.

We conclude that:

**Theorem 1.** *For any $k$ (counting label swaps in a data D), $max\{LR(D, \lambda) : \lambda \in B(\lambda_0, \frac{k}{n})\}$ is attained by swapping the labels in one of the sets listed in $C_U$ and therefore determined by a triplet $(i_1, i_2, i_3) \in T(k, 3)$. Similarly, $min\{LR(D, \lambda) : \lambda \in B(\lambda_0, \frac{k}{n})\}$ is attained by a set in $C_L$ and therefore also determined by some (other) triplet $(i_1, i_2, i_3) \in T(k, 3)$.*

## 3 Results

We now demonstrate the calculation of stability intervals on three different datasets (see Kaplan-Meier curves in Figs 1, 4 and 5). In calculating the interval, we use our efficient LoRSI algorithm, which is considering only a small set of relevant swaps, depending on the value of $\alpha$, as described above. To provide a more complete information on how labeling errors can affect a given dataset, we also present the full $P$-value distribution. In order to do this, we calculate the log-rank $P$-value for all possible swaps according to $\alpha$. The $P$-value distributions for $\alpha = \frac{1}{n}$, pertaining to the $n + 1$ swaps (including the non-swap original data) are depicted in Figure 3B, D and F. In addition, we present, in Figure 3A, C and E, for each dataset, the $P$-value as a function of the time and type of the swapped sample. For the first dataset, we also calculated the interval for 2 and 3 changes (Fig. 6). It should be noted that the calculation of the full distribution introduces a prohibitive time complexity. A more detailed comparison to our efficient approach is given below.

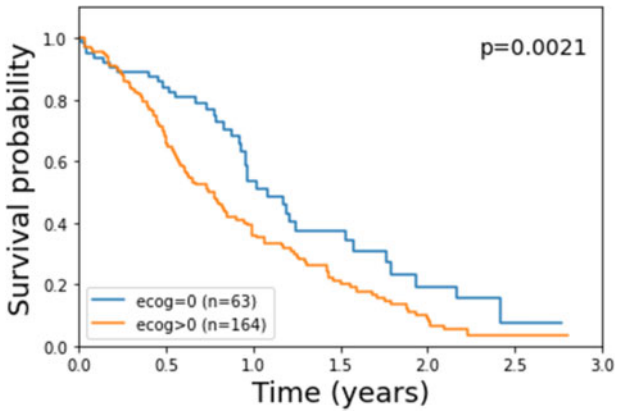


**Fig. 4.** Kaplan Meier curve and log-rank p-value according to ECOG score of patients with advanced colorectal or lung cancer - ECOG=0 Vs ECOG > 0

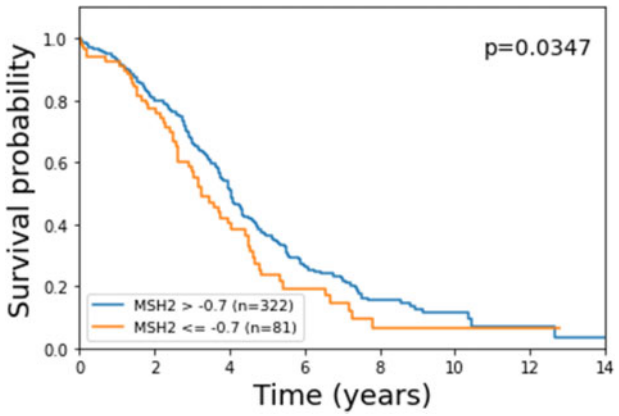


**Fig. 5.** Kaplan Meier curve and log-rank p-value according to expression of the gene MSH2 in ovarian cancer patients - MSH2 expression >-0.7 Vs MSH2 expression ≤-0.7

The first dataset is the MAINZ cohort (Schmidt *et al.*, 2008). We divided the data according to the subtype—Luminal A versus not Luminal A. It is well known that the breast cancer subtype Luminal A has better prognosis than the rest of the subgroups (Fallahpour *et al.*, 2017; Howlader *et al.*, 2018). As expected and as stated in the introduction, a log-rank test demonstrates this difference with $P$-value = 0.014 (see the left panel in Fig. 1). The stability interval calculated given $\alpha = \frac{1}{n}$ & $\delta = 0$ is [0.006, 0.029], see Figure 3A and B. This interval represents a 57% and 107% decrease/increase from the original $P$-value, respectively. We note that $\alpha = \frac{1}{n}$ in this dataset is only 0.5%. Using $\delta = 0.05$, the effect is still dramatic: a 101% increase to the inferred maximum $P$-value (SI = [0.006, 0.0282]). We further investigate the effect of $\alpha = \frac{2}{n}$ and $\alpha = \frac{3}{n}$. The stability interval calculated for $\alpha = \frac{2}{n}$ is [0.0029, 0.055] and when using $\delta = 0.05$, we got $P_U = 0.034$. The stability interval calculated for $\alpha = \frac{3}{n}$ is [0.0013, 0.095] and when using $\delta = 0.05$ we got $P_U = 0.043$. Here, again, we calculated the full $P$-value distribution to provide the complete information (see Fig. 6), a time consuming process. In order to find the stability interval, using the full $P$-value distribution for $\alpha = \frac{k}{n}$, one needs to perform $\sum_{i=1}^{k} \binom{n}{i}$ log-rank calculations. Our LoRSI algorithm needs only $2\binom{k+2}{2}$ such calculations, as described in Section 2. It took 2.5 min to calculate the SI using the full $P$-value distribution, for $\alpha = \frac{2}{n}$, where LoRSI took 0.5 s. For $\alpha = \frac{3}{n}$, the gap is much larger: almost 3 h to calculate the SI using the full $P$-value distribution and only 0.75 s for LoRSI.

Furthermore, setting $\alpha = 0.04$, which represents the error rate according to Ebbert *et al.* (2011), we need to investigate $k = 8$ changes, which is totally impractical. LoRSI will take seconds to do the SI calculation.

The second dataset came from a study that was developed to compare descriptive information from a patient-completed questionnaire to that obtained by the patient's physician Loprinzi *et al.* (1994). All the patients suffered from advanced colorectal or lung cancer. We consider the ECOG score calculated by a physician that assess the patients as a label for assessing survival differences. The 0 score represents fully active, able to carry on all pre-disease activities without restriction. Higher ECOG means less ability to perform usual daily activities, where 5 is the highest score. We divided the data according to ECOG $= 0$ and ECOG $> 0$. The statistical difference in survival between the groups is significant with log-rank $P$-value $= 0.0021$ (Fig. 4). The stability interval calculated given $\alpha = \frac{1}{n}$ & $\delta = 0$ is [0.0004, 0.0079], see in Figure 3C and D. This interval represents a 81% and 276% decrease/increase from the original $P$-value, respectively.

The third and last dataset comes from an investigation of the gene expression in ovarian cancer patients, using the TCGA data (Network *et al.*, 2011). The MSH2 gene was shown to be associated with survival in ovarian cancer (Borcherding *et al.*, 2018). We took the relevant part of the TCGA dataset and split the samples into two groups according the optimal cutoff suggested by Borcherding *et al.* (2018). This cutoff is (standardized) MSH2 expression $>-0.7$ and it yields a log-rank $P$-value of 0.0347 (Fig. 5). The resulting SI at $\alpha = \frac{1}{n}$ is [0.009275, 0.0947], see in Figure 3E and F. Here, the SI represents a 73% decrease from the original $P$-value to the minimum $P$-value and a 273% increase to the maximum $P$-value. This $P_U$ results from swapping only one single sample (the latest censored sample in $C_F$), which is 0.24% of the samples in the cohort. Moreover, increasing $\delta$ to 0.05 will change $P_U$ to 0.06, still a dramatic effect. To obtain $P_U < 0.05$, we need to set $\delta$ to 0.1. The meaning of this result is that 10% of the single sample labeling swaps, applied to a dataset that originally had a significant survival signal, result in a non-significant $P$-value.

## 4 Discussion

In this work, we introduce the novel concept of stability interval for log-rank test. This interval represents the possible effects of perturbing the labels from the original survival analysis data. We show that even a small error rate in the labels can lead to dramatically different statistical conclusions. Our calculated stability interval bounds these differences, thus allowing an assessment of the stability of the statistical test, under labeling errors. We focus on the definition of the stability interval for log-rank and develop an algorithm for efficiently calculating the interval for any $\alpha$.

We present a deterministic approach for addressing the labeling error issue, where we consider all possible label swaps that affect different sample sets representing exactly $\alpha$ fraction of the samples. One can also take a stochastic approach, wherein instances are generated, in which each sample label is swapped with probability $\alpha$. The number of labels actually swapped will then have a Binom$(n, \alpha)$ distribution. Sampling sufficiently many instances, or analytically characterizing the resulting sample space, will lead to a new way of calculating the stability interval from the resulting $P$-value distribution. While in the deterministic approach, we (in effect) assume a uniform error distribution, in this stochastic approach we can, theoretically, use any error distribution. This includes, e.g. models that would assign confidence to individual labels, making swaps less or more likely for individual subjects in the cohort. The study of this interesting and potentially useful extension is a topic for future research. Our approach is also extended to address a confidence parameter $\delta$. Specifically, we find the smallest number $p_U$ for which $LR(D, \lambda) \in [p_L, p_U]$ holds for a $1 - \delta$ fraction of possible labeling changes $\lambda \in B(\lambda_0, \alpha)$. This represents a conservative approach to taking $\delta$ into account. Namely, one that focuses on the desired significance threshold, as may be determined, by the user, in the study design.

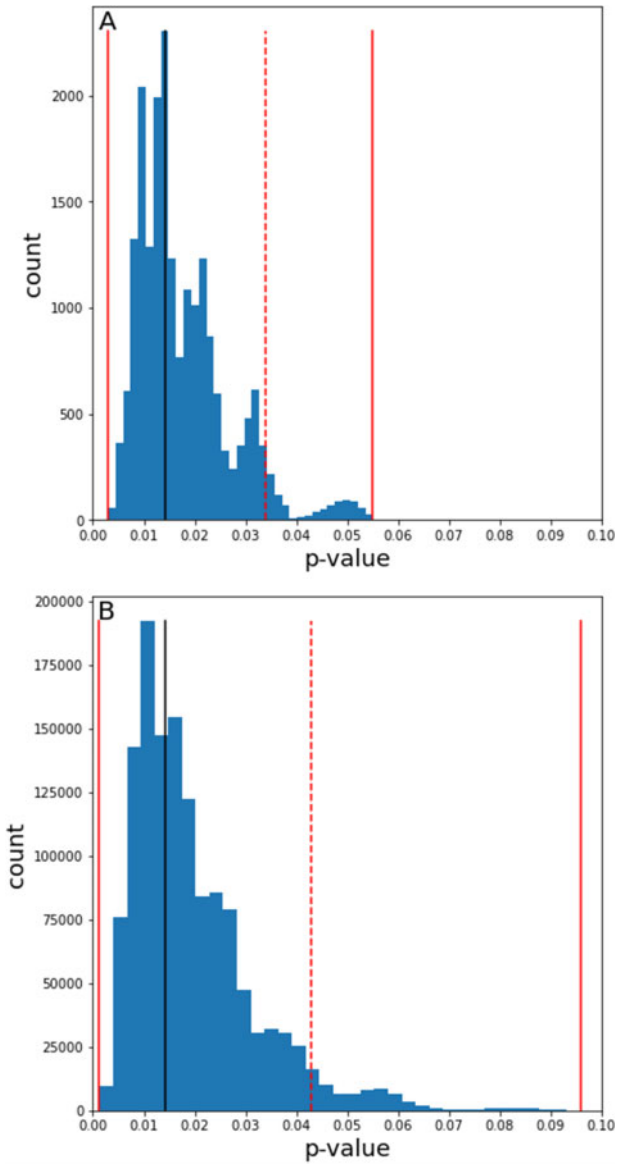


**Fig. 6.** p-value distribution on the MAINZ cohort (Luminal A versus not Luminal A), when $\alpha = \frac{2}{n}$, panel A, and when $\alpha = \frac{3}{n}$, panel B. The black line is the original p-value. The solid red lines are the $P_L$ and $P_U$ when $\delta = 0$. The dashed red line represents the $P_U$ when $\delta = 0.05$

We note that in the proof of our algorithmic approach, we distinguish between working with the CDFs of sums of hypergeometric distributions and working with their standardized versions. Our result, pertaining to how the ends of the log-rank SI can be obtained by calculating the results of $2\binom{k+2}{2}$ sets of $k$ swapped, holds for large $J$s as it requires a normal approximation. If differences in survival are directly assessed against the underlying sum of hypergeometric variables null model, then some of our results hold for any $J$.

We investigated the advantage of using our efficient LoRSI approach as compared to calculating the stability interval by generating the full $P$-value distribution. While LoRSI performs $\Theta(k^2)$ log-rank calculations to address $k$ changes, the exhaustive approach takes $\Theta\binom{n}{k}$ such calculations. This complexity gap leads to seconds versus hours difference for small values of $k$ and to LoRSI being the only practical approach in higher values.

We provide a Python implementation of the LoRSI algorithm. Current work focuses on the development of more efficient and user

friendly implementations of the methods described herein as well as on visualization tools. All will be made available through future releases. We hope that such efforts will make statistical stability analysis more accessible and useful for the community.

## Acknowledgements

We thank the Technion Computer Science Department and the School of Computer Science at IDC Herzliya, for their support of the project. We thank the Yakhini Research Group for important discussions and input. We thank Xavier Tekpli for important discussion and insights.

## Funding

This work was supported by the European Union's Horizon 2020 Research and Innovation Program under RESCUER, GA No. [847912].

*Conflict of Interest*: none declared.

## References

Borcherding,N. *et al.* (2018) TRGAted: a web tool for survival analysis using protein data in the cancer genome atlas. *F1000Res.*, **7**, 1235.

Cover,T.M. (1999) *Elements of Information Theory*. John Wiley & Sons, NJ, USA.

Ebbert,M.T. *et al.* (2011) Characterization of uncertainty in the classification of multivariate assays: application to pam50 centroid-based genomic predictors for breast cancer treatment plans. *J. Clin. Bioinform.*, **1**, 37.

Elmore,J.G. *et al.* (2017) Pathologists' diagnosis of invasive melanoma and melanocytic proliferations: observer accuracy and reproducibility study. *BMJ*, **357**, j2813.

Fallahpour,S. *et al.* (2017) Breast cancer survival by molecular subtype: a population-based analysis of cancer registry data. *CMAJ Open*, **5**, E734–E739.

Galili,B. *et al.* (2021) Efficient gene expression signature for a breast cancer immuno-subtype. *PLoS One*, **16**, e0245215.

Ha,R. *et al.* (2019) Predicting breast cancer molecular subtype with MRI dataset utilizing convolutional neural network algorithm. *J. Digit. Imaging*, **32**, 276–282.

Heimann,G. and Neuhaus,G. (1998) Permutational distribution of the log-rank statistic under random censorship with applications to carcinogenicity assays. *Biometrics*, **54**, 168–184.

Hothorn,T. and Lausen,B. (2003) On the exact distribution of maximally selected rank statistics. *Comput. Stat. Data Anal.*, **43**, 121–137.

Hougaard,P. (1995) Frailty models for survival data. *Lifetime Data Anal.*, **1**, 255–273.

Howlader,N. *et al.* (2018) Differences in breast cancer survival by molecular subtypes in the united states. *Cancer Epidemiol. Biomarkers Prev.*, **27**, 619–626.

Islam,M.M. *et al.* (2020) An integrative deep learning framework for classifying molecular subtypes of breast cancer. *Comput. Struct. Biotechnol. J.*, **18**, 2185–2199.

Jaber,M.I. *et al.* (2020) A deep learning image-based intrinsic molecular subtype classifier of breast tumors reveals tumor heterogeneity that may affect survival. *Breast Cancer Res.*, **22**, 12.

Jackson,S.L. *et al.* (2017) Diagnostic reproducibility: what happens when the same pathologist interprets the same breast biopsy specimen at two points in time? *Ann. Surg. Oncol.*, **24**, 1234–1241.

Kleinbaum,D.G. and Klein,M. (2012) *Survival Analysis*. Springer, New York.

Levy-Jurgenson,A. *et al.* (2020) Spatial transcriptomics inferred from pathology whole-slide images links tumor heterogeneity to survival in breast and lung cancer. *Sci. Rep.*, **10**, 18802.

Lindeberg,J.W. (1922) Eine neue herleitung des exponentialgesetzes in der wahrscheinlichkeitsrechnung. *Math. Z.*, **15**, 211–225.

Loprinzi,C.L. *et al.* (1994) Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. *J. Clin. Oncol.*, **12**, 601–607.

Mantel,N. (1966) Evaluation of survival data and two new rank order statistics arising in its consideration. *Cancer Chemother. Rep.*, **50**, 163–170.

Network,C.G.A.R. *et al.* (2011) Integrated genomic analyses of ovarian carcinoma. *Nature*, **474**, 609.

Pitt,B. *et al.* (1999) The effect of spironolactone on morbidity and mortality in patients with severe heart failure. *N. Engl. J. Med.*, **341**, 709–717.

Schmidt,M. *et al.* (2008) The humoral immune system has a key prognostic impact in node-negative breast cancer. *Cancer Res.*, **68**, 5405–5413.

Tourneau,C.L. *et al.*; SHIVA investigators. (2015) Molecularly targeted therapy based on tumour molecular profiling versus conventional therapy for advanced cancer (SHIVA): a multicentre, open-label, proof-of-concept, randomised, controlled phase 2 trial. *Lancet Oncol.*, **16**, 1324–1334.

Vandin,F. *et al.* (2015) Accurate computation of survival statistics in genome-wide studies. *PLoS Comput. Biol.*, **11**, e1004071.

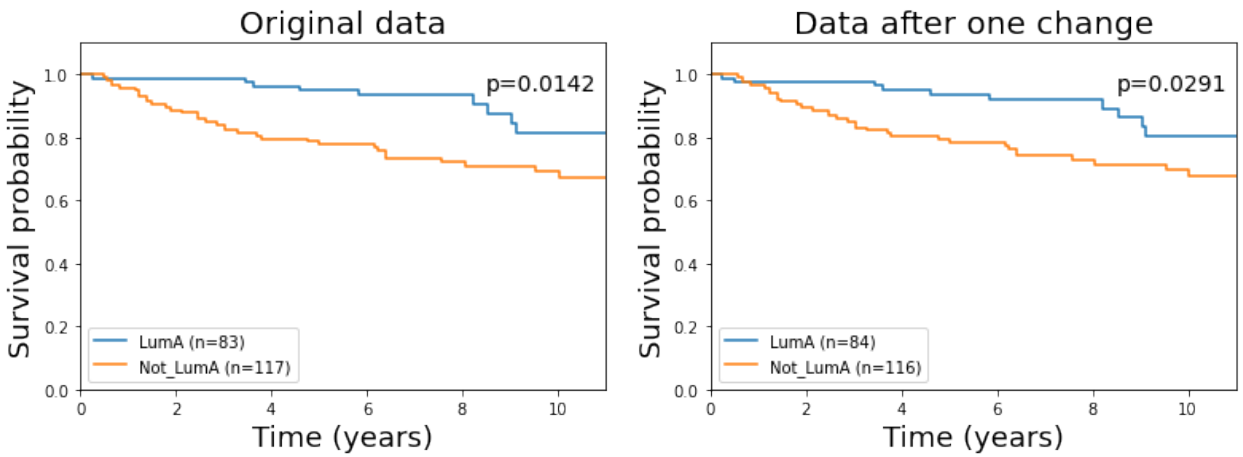


Figure 6.1: Kaplan Meier curve and log-rank p-value on the MAINZ cohort - Luminal A Vs not Luminal A. On the left the original data and on the right the data after one change.

# Part III

# Summary And Discussion

## 6.1 Summary

This work has two parts.

In the first part we suggest a statistically motivated design of a secure MPC Logrank Test protocol. If the goal is to satisfy MPC security, and the problem is the complexity of the Logrank process, the problem can be dramatically simplified by minimizing the number of operations on the ciphertext.

In CoPPSA the only operations done on encrypted data are sums and multiplications of integers, and the amount of messages sent is linearly proportional to the number of parties. Moreover: the complexity of the protocol is detached from the dataset size. Comparing to von Maltitz et al solution, the advantages of our protocol are simplicity and flexibility, which leads to significantly higher performance on large datasets.

CoPPSA is simple and modular. It can be implemented based on a variety of known MPC tools like BGW, Homomorphic Encryption, or others. The security properties are inherited directly from the chosen MPC tool. Potential users can therefore implement an MPC version of Logrank according to their adversary model and complexity requirements.

CoPPSA has another significant advantage over the alternative solutions: it encourages the parties to use their true values instead of false values (see chapter 5). Consider a case in which some party decides to "lie" , that is to provide values which are not coherent with it's actual dataset. By doing so the party risks a loss of the required Logrank result. Assuming that obtaining the Logrank result is more desirable than prevent it from the other parties, this is a strong incentive to comply.

It is important to note, however, that CoPPSA calculates a slightly different function than the original Logrank, due to a loss of information. The distance between the calculations decay as the merged dataset's size increases (see section 4.1). By proving CoPPSA converge to standard normal distribution we showed that the result our protocol provides is equivalent to the standard Logrank result, and therefore reliable.

In the second part we demonstrate the uncertainty that arises from labeling errors. Given survival data with $n$ samples and an error rate, $\alpha$, we define a stability interval and a procedure that calculates a stability interval for the log-rank test.

## 6.2 Future Directions

As a proof of concept, we provide a single threaded simulation of CoPPSA. In the implementation given at https://github.com/asamohi/Logrank_submodule.git . Optional extension for this project can be implement the protocol as a multi-process web application with a user interface.

In our simulation we assume that the adversary does not corrupt both servers at the same run. This assumption is taken for simplicity, since this project is not focused on the cryptographic aspects. This assumption is not mandatory, the protocol can be implemented according to any adversary model.

In our research we provided a proof that CoPPSA converges to standard normal distribution. We also provided an empirical results showing that for database large enough $Z* \sim Z$. Optional progress can be providing a full analytic proof that $Z^* \sim Z$. First step would be to identify the characters of database partitions that leads to radical results

$$RadicalPartition = argmax_{partitions}|Z^* - Z|$$

The next step would be to analyze the probability for such radical cases.

In chapter 5 we present a new definition for a function with an incentive for veracity. We are currently developing this subject as a separate research.

# Bibliography


[1] Eran Eden et al. “GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists”. In: *BMC Bioinformatics, volume 10* (2009).

[2] Anna V Ivshina et al. “Genetic reclassification of histologic grade delineates new clinical subtypes of breast cancer”. In: *Cancer Res, volume 66* (2006).

[3] Lao H Saal et al. “The Sweden Cancerome Analysis Network - Breast (SCAN-B) Initiative: a large-scale multicenter infrastructure towards implementation of breast cancer genomic analyses in the clinical routine”. In: *Genome Medicine, volume 7* (2015).

[4] Sherene Loi et al. “Definition of clinically distinct molecular subtypes in estrogen receptor-positive breast carcinomas through genomic grade”. In: *Journal of Clinical Oncology, volume 25* (2007).

[5] Tingting Chen and Sheng Zhong. “Privacy-Preserving Models for Comparing Survival Curves Using the Logrank Test”. In: *Computer Methods and Programs in Biomedicine, volume 104* (2011).

[6] David Collett. “Modelling survival data in medical research”. In: Chapman and Hall, 1994. ISBN: 1439856788.

[7] Richard Ha et al. “Predicting Breast Cancer Molecular Subtype with MRI Dataset Utilizing Convolutional Neural Network Algorithm”. In: *Journal of Digital Imaging, volume 32* (2019).

[8] W. L. Harkness. “Properties of the Extended Hypergeometric Distribution”. In: *Ann. Math. Statist. volume 36* (1965).

[9] Md. Mohaiminul Islam et al. “An integrative deep learning framework for classifying molecular subtypes of breast cancer”. In: *Computational and Structural Biotechnology Journal, volume 18* (2020).

[10] Mustafa I Jaber et al. “A deep learning image-based intrinsic molecular subtype classifier of breast tumors reveals tumor heterogeneity that may affect survival”. In: *Breast Cancer Research, volume 22* (2020).

[11] Yehuda Lindell. “Tutorials On The Foundations Of Cryptography”. In: Springer Nature, 2017. ISBN: 9783319570488.

[12] A.Wigderson M.Ben-Or S.Goldwasser. “Completeness theorems for non-cryptographic fault-tolerant distributed computation”. In: *Proceedings of the 20th Annual Symposium on the Theory of Computing (STOC’88), pages 1-10* (1988).

[13] Marcel von Maltitz et al. “A Privacy-Preserving Log-Rank Test for the Kaplan-Meier Estimator With Secure Multiparty Computation: Algorithm Development and Validation”. In: *JMIR Medical Informatics, volume 9* (2021).

[14] Nathan Mantel. “Evaluation of survival data and two new rank orders order statistics arising in its consideration”. In: *Cancer chemotherapy reports, volume 50* (1966).

[15] Michael Monagan. “Maximal Quotient Rational Reconstruction: An Almost Optimal Algorithm for Rational Reconstruction”. In: *Proceedings of the 2004 international symposium on Symbolic and algebraic computation* (2004).

[16] J. E. Mymann. “On the Probability that k Positive Integers are Relatively Prime”. In: *Journal of Number Theory, volume 4* (1972).

[17] Thong T. Nguyen and Siu Cheung Hui. “Privacy-Preserving Mechanisms for Parametric Survival Analysis with Weibull Distribution”. In: *IEEE Trustcom/BigDataSE/ICESS* (2017).

[18] Michael O. Rabin. “How to Exchange Secrets with Oblivious Transfer”. In: *Technical Report TR-81, Aiken Computation Lab, Harvard University* (1981).

[19] Marcus Schmidt et al. “The humoral immune system has a key prognostic impact in node-negative breast cancer”. In: *Cancer research, volume 68* (2008).

[20] Yoav Shoham and Moshe Tennenholtz. “Non-cooperative computation: Boolean functions with correctness and exclusivity”. In: *Theoretical Computer Science, volume 343* (2005).

[21] Paul S. Wang. “A p-adic algorithm for univariate partial fractions”. In: *Proceedings of the fourth ACM symposium on Symbolic and algebraic computation (SYMSAC’81)* (1981).

[22] Andrew Chi-Chih Yao. “How to generate and exchange secrets”. In: *27th Annual Symposium on Foundations of Computer Science (SFCS1986), IEEE* (1986).

# תקציר

מחקר זה עוסק במבחן לוגרנק (Logrank Test) - אלגוריתם ממשפחת האלגוריתמים הסטטיסטיים המשמשים לניתוח שרידות. חיבור זה כולל שני חלקים.

בחלק הראשון נציג פרוטוקול לחישוב משותף בטוח רב-משתתפים (multi party computation) של קירוב תוצאת מבחן הלוגרנק. נוכיח את נכונות הפרוטוקול ונראה שניתן לממשו בכל רמת בטיחות קריפטוגרפית נדרשת וכנגד כל מודל של תוקף, וזאת על בסיס שימוש בכלים קריפטוגרפיים קיימים. נראה את אמינות הקירוב המחושב ביחס לתוצאת מבחן הלוגרנק המקורי. בהמשך נציג אופן טיפול חדש בבעיית קלט שיקרי (false input) וכן גרסה ספציפית של הפרוטוקול שמספקת הגנה חלקית כנגד מתקפה זו.

בחלק השני ננתח את השפעתן של טעויות תיוג בקבוצת הנתונים על תוצאת המבחן (Logrank). נציג פתרון חדש להתמודדות עם הבעיה ע״י אלגוריתם המחשב תחום יציבות קטן ככל האפשר (stability interval) לתוצאת המבחן בהתחשב בשיעור שגיאות אפשרי בנתונים. מאמר אודות הפתרון המוצע פורסם במגזין Bioinformatics ביולי 2021 והוא מצורף לחיבור זה.

עבודה זו בוצעה בהדרכתם של פרופ' זוהר יכיני מבי"ס אפי ארזי למדעי המחשב, אוניברסיטת רייכמן, ודר' עדי עקביה מהמחלקה למדעי המחשב, אוניברסיטת חיפה.

בית ספר אפי ארזי
למדעי המחשב
הבינתחומי
הרצליה

המרכז הבינתחומי בהרצליה
בית-ספר אפי ארזי למדעי המחשב
התכנית לתואר שני (M.Sc.) - מסלול מחקרי

# אלגוריתם Logrank:
# ניתוח יציבות וחישוב בטוח מרובה משתתפים

מאת
**ענת שמוחי**

עבודת תזה המוגשת כחלק מהדרישות לשם קבלת תואר מוסמך M.Sc.
במסלול המחקרי בבית ספר אפי ארזי למדעי המחשב, אוניברסיטת רייכמן

יוני 2022