**HOMEWORK 1**

**1. Basic Notions in Statistics**

* **Population**: In statistics, a population refers to the complete set of individuals, items, or events of interest that researchers want to study. It can be finite (e.g., the population of a country) or infinite (e.g., possible outcomes of rolling a die). Understanding the population helps define the scope of a study and the data collected.
* **Statistical Units**: These are individual elements or entities in the population, such as a person, an object, or a specific observation. A statistical unit is the smallest entity that can be measured for statistical purposes.
* **Distribution**: A distribution describes how values of a random variable are spread or distributed. This can be summarized using probability distributions (for random variables) or frequency distributions (for actual data). Common types include normal, binomial, and Poisson distributions, each providing insights into the probability and occurrence of various outcomes.

**2. Notion of Average and Computational Problems**

* **Average**: An average (mean) is a measure of central tendency that summarizes a set of data by calculating the sum of all values and dividing by the number of values. It provides a quick way to understand the "typical" value in a dataset.
* **Floating Point Representation**: Computers store numbers using a finite number of bits, which leads to approximation errors, especially for very small or very large numbers. In floating-point arithmetic, numbers are represented as fractions and exponents, but this can cause issues such as:
  + **Errors**: Precision errors occur because floating-point numbers can only represent a limited set of values, so some numbers are rounded, causing small inaccuracies in calculations.
  + **Catastrophic Cancellation**: This occurs when subtracting two nearly equal numbers, leading to a significant loss of precision. Even a minor floating-point error can dramatically affect the result.
* **Knuth’s Numerical Solutions**: Donald Knuth proposed various numerical methods to minimize errors in floating-point arithmetic. By rethinking the order of operations and using algorithms that reduce rounding errors, numerical stability can be improved, ensuring more accurate results even with floating-point limitations.

**Key Parts of the Code**

1. **Input Parameters**:
   * **Number of Servers** (n): This represents how many servers are in the system, each of which can either be penetrated or not by an attacker.
   * **Number of Attackers** (m): The number of attackers who will attempt to penetrate the servers.
   * **Penetration Probability** (p): The probability that each attacker will successfully penetrate a server. This is a random chance for each server, and it varies slightly for each attacker to ensure that the lines do not overlap too much.

A screenshot of a computer code

Description automatically generated

1. **Penetration Simulation**:
   * The core of the simulation involves each attacker attempting to penetrate each server. For each attacker and each server, the code generates a random value and compares it with the penetration probability.
   * If the attack is successful, a visual jump is made on the Y-axis of the graph. If not, the line stays flat (at the "No Penetration" level).
   * A screenshot of a computer program

     Description automatically generated
2. **Dynamic Line Colors and Styles**:
   * Attackers are visually differentiated by assigning each one a unique color and line style (solid, dashed, dotted, etc.). This allows us to see the path each attacker takes across the servers.
3. **Markers for Penetration**:
   * A small circle is drawn at each server point where an attacker successfully penetrates, further emphasizing the success of the attack.
4. **Dynamic Legend**:
   * The legend on the right side of the graph updates dynamically based on the number of attackers, showing which color represents each attacker.

A screenshot of a computer code

Description automatically generated

1. **Graphical Output**:
   * The X-axis represents the servers, while the Y-axis indicates whether penetration was successful or not. The graph uses different lines and markers to show the path each attacker took and the outcome of their attempts to penetrate the servers.

**Explanation of Results Using the Provided Pictures**

A screenshot of a graph

Description automatically generated

**Description:** The graph shows the results of 4 attackers attempting to penetrate 7 servers with a 0.2 penetration probability (low probability).

**Observations:**

* **Server 1**: All attackers successfully penetrated the first server, as indicated by the various lines starting from the bottom and jumping up.
* **Server 2-7:** The results vary more. For example, Attacker 1 (Red) successfully penetrated Servers 1 and 7 but failed to penetrate others. Attacker 2 (Green) managed to penetrate Servers 1, 3, and 5 but failed on Servers 2 and 6.
* **Result:** The overall results reflect the low penetration probability (0.2), as many attackers fail to penetrate servers, shown by the flat or downward lines between server points.

A screenshot of a graph

Description automatically generated

 **Description**: In this graph, we have 5 servers, 3 attackers, and a higher penetration probability (0.6).

 **Observations**:

* **Higher Penetration Success**: With a higher probability of penetration (0.6), more attackers successfully penetrate more servers. **Attacker 1** (Red) penetrated Servers 1, 2, and 5, while **Attacker 2** (Green) successfully penetrated Servers 1, 3, and 5.
* **Overlap**: Due to the higher success rate, we see more overlaps between the attackers' lines. For example, multiple attackers penetrated Server 1, leading to overlapping lines and markers on the graph.
* **Result**: The higher penetration probability results in more successful attacks across servers, as reflected by the more frequent upward jumps in the lines.

**Conclusion**

The simulation demonstrates how various factors—such as the number of servers, number of attackers, and penetration probability—affect the outcome of penetration tests on servers. The graphical output clearly shows the path of each attacker, using distinct colors and line styles to differentiate between them. By adjusting the penetration probability, we can simulate different attack scenarios, from a highly secure environment (low penetration probability) to one where attackers have a greater chance of success (higher penetration probability).

This simulation tool can be used to analyze the security robustness of a system by observing how different attackers succeed or fail at penetrating various server setups, offering insights into potential vulnerabilities.