Theoretical Notions

In this section, we will find some notions relating to statistics.

Statistics is a branch of mathematics that makes it possible to study a set of scientific methods aimed at quantitative and qualitative knowledge of collective phenomena through the collection, ordering, synthesis and analysis of data.

Definition 1 (Population)

A complete set of elements or individuals that possess a common characteristic and on which an analysis or measurement is to be made. We can represent the population by a set in which statistical units are found.

Definition 2 (Statistical Units)

Individual elements that make up a population and on which data are collected in a statistical study. Each statistical unit represents a specific case that can be observed, measured or recorded.

When collecting data on statistical units, each statistical unit is associated with specific values for the variables of interest. These values can be represented as an **instance**: $(x1, x2, \ldots, xk)$.

If we collect data from n statistical units, we obtain a dataset of instances. Each instance corresponds to a statistical unit and contains values of the measured variables.

Practical example of a dataset of n persons:

$\text{Age}(x_1)$	$\text{Weight}(x_2)$	$\text{Height}(x_3)$
30	70	175
25	60	160
40	80	180
$\vdots$	$\vdots$	$\vdots$

Definition 3 (Distribution)

Representation of the frequency with which each value (or range of values) appears among the statistical units. It provides information on:

1. **Frequency**: how many elements in the population take a certain value or fall within a certain range of values.
2. **Shape**: the shape of the distribution can reveal trends, symmetries, asymmetries or the presence of outliers.
3. **Centre and Dispersion**: the distribution provides information about the centre (e.g. mean or median) and dispersion (e.g. variance or standard deviation) of the values of statistical units.

Definition 4 (Average)

Represents the mean value of a set of data.

There are several types of average, but the most common are the **arithmetic average**, the **weighted average** and the **geometric average**.

1. **ArithmeticAverage**:

$$ArithmeticAverage = \frac{\sum_{i=1}^{n} x_i}{n}$$

where:

- x_i: i-th element of an instance.
- n: number of elements of an instance.

  Demonstration:

  - Given a point μ within a distribution, we know that that point is the mean if and only if the sum of the elements to its left is equal to the sum of the elements to its right ($sum_{left} = sum_{rigth}$).
  - Arbitrarily taking two points x_i and x_j to the left and right of μ respectively, we can denote the distance between them as: $(\mu - x_i)$ and $(x_j - \mu)$.
  - If we call k the index of the element equal to μ, we can arrive at writing:
    - $sum_{left} = \sum_{i=1}^{k} (\mu - x_i)$
    - $sum_{right} = \sum_{j=k+1}^{n} (x_j - \mu)$
  - We therefore obtain:

$$\sum_{i=1}^{k}(\mu - x_i) = \sum_{j=k+1}^{n} (x_j - \mu)$$

$$\Rightarrow \sum_{j=k+1}^{n} (x_j - \mu) - \sum_{i=1}^{k}(\mu - x_i) = 0$$

$$\Rightarrow \sum_{j=k+1}^{n} (x_j - \mu) + \sum_{i=1}^{k}(x_i - \mu) = 0$$

$$\sum_{i=1}^{n}(x_i - \mu) = 0 \Rightarrow -(\mu \cdot n) + \sum_{i=1}^{n}(x_i) = 0 \Rightarrow \mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

2. **Weighted Average**: a **weight** (w_j) is added to each single element x_i:

$$WeightedAverage = \frac{\sum_{i=1}^{n} x_i \cdot w_i}{\sum_{i=1}^{n} w_i}$$

3. **Geometric Mean**: average of values that are expressed as ratios or percentages:

$$GeometricAverage = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

In Computer Science:

With regard to average, the recursive formula is used in computer science because it is computationally less expensive:

Recursive Formula of Average:

$$Average_n = Average_{n-1} + \frac{x_n - Average_{n-1}}{n}$$

Demonstration:

1. **We know that**:
   $S_n = S_{n-1} + x_n$
   where:
   - S_{n-1}: sum of the first $n-1$ values.

- **The arithmetic mean of the first $n-1$ values** is:
  $Average_{n-1} = \frac{S_{n-1}}{n-1}$ From which we obtain S_{n-1}:
  $S_{n-1} = (n-1) \cdot Average_{n-1}$
- **We replace S_{n-1} in the formula for S_n**:
  $S_n = (n-1) \cdot \text{Average}_{n-1} + x_n$
- **Now let's calculate the new average**:
  $Average_n = \frac{S_n}{n} = \frac{(n-1) \cdot \text{Average}_{n-1} + x_n}{n}$
- **Separate terms**:
  $Average_n = \frac{(n-1) \cdot \text{Average}_{n-1}}{n} + \frac{x_n}{n}$
- **We can rewrite the first part**:
  $Average_n = \text{Average}_{n-1} \cdot \frac{n-1}{n} + \frac{x_n}{n}$
- **Finally, we express the difference between the new value and the previous average**:
  $Average_n = \text{Average}_{n-1} + \frac{x_n - \text{Average}_{n-1}}{n}$

Floating point representation

The **floating point representation** is a method of representing real numbers in computers so that they can be used in mathematical calculations. This representation is particularly useful for handling very large numbers, very small numbers, and fractions, allowing a balance to be maintained between precision and representational capacity.

Structure of the Floating Point Representation

The floating-point representation of a number can be seen as a way of expressing the number in the form:

$$number = \text{sign} \times \text{mantissa} \times \text{base}^{\text{exponent}}$$

Where:

- **Sign**: indicates whether the number is positive or negative.
- **Mantissa**: represents the significant value of the number. It is a fractional number which, in many formats, is normalised so that the first digit (to the left of the decimal point) is non-zero.
- **Base**: in most implementations, the base is 2 (binary system), but can also be 10 (decimal system) in other representations.
- **Exponent**: determines the magnitude of the number, indicating by how much the mantissa must be multiplied to obtain the final value.

IEEE 754 Standard

The most common floating-point representation format is defined by the **IEEE 754** standard, which specifies various formats for floating-point numbers. The best-known formats include:

1. **Single Precision (32 bits)**:
   - **1 bit for sign**: indicates whether the number is positive (0) or negative (1).
   - **8 bits for exponent**: allows exponents ranging from -126 to 127 to be represented.
   - **23 bits for the mantissa**: includes the fractional part of the number.

   The general representation for single precision numbers is:
   $number = (-1)^{\text{sign}} \times (1.mantissa) \times 2^{(exponent-127)}$
2. **Double Accuracy (64 bits)**:
   - **1 bit for the sign**.
   - **11 bits for the exponent**: allows exponents ranging from -1022 to 1023 to be represented.
   - **52 bits for the mantissa**.

   The general representation for double precision numbers is:
   $\text{number} = (-1)^{\text{sign}} \times (1.mantissa) \times 2^{(exponent-1023)}$

Representation Errors

Representation errors occur when a real number cannot be represented exactly in floating-point form. This can happen for several reasons:

1. **Precision Limits**: Since floating-point numbers have a finite precision, some real values cannot be expressed exactly. For example, the number 0.1 does not have an exact representation in base 2. When converted, the computer stores an approximation of this number.
2. **Rounding**: when a number is represented as an approximation, rounding errors can occur. This occurs when a value must be truncated or rounded to fit the floating-point format. For example, if an attempt is made to store $3.141592653589793\ (\pi)$ in a single-precision format, the computer may only store an approximation, leading to a rounding error.
3. **Overflow and Underflow**:

- **Overflow**: occurs when a calculated number exceeds the maximum value that can be represented in floating point, resulting in an infinite value.
- **Underflow**: occurs when a calculated number is too close to zero to be represented correctly, resulting in zero.

Catastrophic Errors

Catastrophic errors occur in mathematical operations in which rounding errors add up to cause a significant loss of precision in the final result. These errors can occur in various contexts, including:

1. **Subtraction of Similar Numbers**: When subtracting two similar numbers (e.g. 1.0000001 and 1.0000000), the difference is small compared to the original values. This can lead to a catastrophic error, as the representation error in each number becomes relatively large compared to the result:

   Example: $(1.0000001 - 1.0000000) = 0.0000001$.
   If the original numbers have a representation error, the final result may be inaccurate.
2. **Summation of Numbers with Very Different Orders of Magnitude**: when summing numbers of very different orders of magnitude, the representation error of the smaller number can be neglected, affecting the accuracy of the result. For example, when adding 1.0×10^{20} and 1.0: $1.0 \times 10^{20} + 1.0 \approx 1.0 \times 10^{20}$
   Here, the addition of 1.0 does not affect the value of 1.0×10^{20} due to the limited precision.

Error Mitigation

To reduce representation errors and catastrophic errors, several strategies can be adopted:

1. **Use of Stable Algorithms**: use numerical algorithms that minimise rounding errors, such as stable sums or controlled error formulas.
2. **Double Precision Formats**: use double precision formats (64 bits) when higher precision is required, even if this entails a higher computational cost.

3. **Calculation Reformulation**: restructure mathematical expressions to avoid subtraction of similar numbers or combinations of numbers with very different orders of magnitude.

Knuth's Strategy for Avoiding Catastrophic Errors

Donald Knuth has proposed several strategies and approaches to handle representation errors and catastrophic errors in computation, especially in numerical contexts. One of the best known strategies he suggested is the use of a method known as **"Arithmetic of Rational Numbers“** or **”Exact Arithmetic"** to reduce rounding errors.

1. **Use of Rational Numbers**
   Knuth suggests using rational numbers (fractions) rather than floating-point numbers to represent values that may be subject to catastrophic errors. The idea is that, by using rational numbers, rounding errors can be avoided because rational numbers can be represented exactly as a pair of integers $(numerator, denominator)$.

For example, instead of representing the number 0.1 in floating point, the rational representation $\frac{1}{10}$ can be used. This approach allows arithmetic operations to be performed without losing precision due to representation errors.

2. **Reformulation of Operations**
   Knuth also emphasises the reformulation of operations to minimise the risk of catastrophic errors. For example, when performing subtraction between similar numbers, it is advisable to reformulate the calculations so that the operation reduces the loss of precision.
3. **Stable Algorithms**
   Knuth recommends using stable numerical algorithms, which are designed to reduce the propagation of rounding errors. This includes techniques such as:

- **Kahan Summing**: a method for summing a series of numbers that tracks rounding errors and corrects them during the summing process.
- **Arbitrary Precision Representation**: use data structures that support arbitrary precision to avoid loss of information in complex calculations.

4. **Error Control**.
   Knuth also spoke about the importance of implementing error control mechanisms in programmes. This may include:

- **Verification of Overflow/Underflow Conditions**: monitor the conditions under which overflow or underflow occurs and handle them appropriately.
- **Validity Testing**: perform checks to verify the correctness of results, e.g. by comparing the results obtained with expected results or with calculations performed in a higher precision format.