Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Experimental Methods in Computer Science

DEI-FCTUC, 2019/2020

This assignment is a first (and simple) exercise of designing an experiment. There are many types of experiments and the term "design of experiments" (often referred as experimental design as well) is used for the process of defining and planning experiments in such a way that the date obtained can be analyzed to draw valid and objective conclusions.

The topic of designing experiments of different types and for different purposes is addressed in detail in the lectures (T classes). In any case, the next paragraphs present a brief overview before presenting the assignment purpose.

A classic experiment includes the following elements (you can take them as basic steps for the purpose of this assignment):

1. Problem statement (or research question)
2. Identify variables
3. Generate hypothesis
4. Define the experimental setup/scenario
5. Develop the necessary tools and procedures for the experiment
6. Run the experiments and collect the data/measurements
7. Perform data analysis
8. Draw conclusions (and often go back to the beginning and reformulate the problem statement or test a different hypothesis)

The formulation of sound problem statements (or research questions) is not easy and this is particularly true for the computer science field. A good (i.e., relevant problem statement) should be focused enough to allow the clear identification of the variables of the problem but, at the same time, should be sufficiently open to allow different hypothesis to answer the problem/question.

An example of formulation of a possible problem statement is:

How does the setting (noise, temperature, music, open space, etc.) of the room used by programmers affect the number of bugs found by test suits in the modules produced by such programmers?

The effect we want to measure (dependent variable) is the number of bugs found and the factors (independent variables) are the different room settings and situations. It is assumed that there is a set of conditions that remain stable (e.g., type of programing language, complexity of the modules, etc.).

A possible hypothesis for the problem statement mentioned above is that programmers tend to make fewer bugs if they listen classic music at a comfortable volume while programming. In this case, the hypothesis is directional (i.e., establish a direction for the dependency between the dependent variable and the independent variable) but the hypothesis could simply state that music influences the number of bugs (non-directional hypothesis).

Even in a simple example like this, it is easy to understand the difficulties associated to the correct validation of hypothesis. For example, it is necessary to perform experiments with a representative group of programmers, use a variety of software under development, assure that the conditions of the different experiment runs remain stable, etc. The analysis of the results in order to check if their distribution fit the hypothesis testing procedure is often quite difficult as well, often requiring complex statistic testing techniques.

1. Goals, assumptions and recommendations

The purpose of the assignment is to design a set of experiments and execute all the steps to evaluate the following general problem statement:

<u>How are sorting algorithms affected by memory faults?</u>

Memory faults are more common that you might think. Standard error rates for memory devices are close to 5000 failures in time per Mbit. This means that a memory chip of 1 Mbit is likely to experience 5000 faults every 10^9 hours of use[1]. Sorting algorithms may be strongly affected by memory faults during their execution, since they may take the wrong path after reading corrupted values and propagate errors throughout their execution.

The goal of this work is to understand how different sorting algorithms are affected by memory faults. In this work, you are considering a particular memory fault model on the address decoder: assume, with a given probability, that a comparison between two values in the array to be sorted is faulty, that is, if A is being compared to B and $A < B$, the outcome is that $A \geq B$. This corresponds to a faulty access to a particular memory location where rather than accessing B, it accesses C, where $A \geq C$. Under this memory fault model, the sorting algorithm may not output a sorted array.

The following assumptions and recommendations should be taken into account:

- Consider sorting algorithms that sort an array of integers in non-decreasing order and that are based on pairwise comparisons, such as the quicksort, mergesort, insertionsort and bubblesort. The implementations of the four sorting algorithms above are available at Inforestudante (read the README.txt in the zip file).

1 In fact, this is no longer a problem. As an optional homework, investigate the techniques that are currently available to detect and correct memory faults.

- We *simulate* the memory fault model explicitly in the code of each sorting algorithm, that is, whenever a comparison is performed between two values in the array, there is a probability that a failure occurs *and* that the outcome of the comparison is wrong. You are free to consider other sorting algorithms as well and other fault models.
- Each of the four implementations receives a file from stdin as input. This file contains a single line. It starts with a real number between 0 and 1, which corresponds to the probability of a fault to occur as above. Then, it is followed by a positive integer, which indicates the size of the array to be sorted ($n \leq 10000$). Then, a list of n positive integers follows, which corresponds to the content of the array.
- In inforestudante, there is a file named gen.py written in python that allows you to generate input files as explained above. You can define the probability of a fault to occur as well as the size and the range of values of the array. In the current version, the array is generated according to an uniform distribution. You can modify the file to consider other distributions.
- Each implementation outputs the following information to stdout, separated by a line break:
  1) The original array;
  2) The array processed by the sorting algorithm under the memory fault model;
  3) The array sorted correctly;
  4) The size of the largest sorted subarray in 2) .

  Note that 4) is a *rough* indicator of the quality of the array produced under the memory fault model. What other indicators could be used? You are free to modify the code to obtain other indicators.
- Note that the PC where you are considering running the experiments may suffer from some real memory faults! Think about ways of overcoming this difficulty.
- The experiments can be repeated using slightly different conditions in order to understand better the situation and allow the generalization of the results. It is open to you to explore this possibility.

2. Outcome

There are three milestones, each of which covers a particular technique discussed throughout the course:

1. *Exploratory data analysis*
2. *Hypothesis testing*
3. *Regression analysis*

The goal of each milestone is to design the experiment, collect the data, and use the corresponding data analysis technique to draw conclusions about the problem statement. The outcome of each milestone is a written report (PDF file, maximum 8 single-column pages, including cover, references and appendices). The report should describe the steps of the design of the experiment with enough detail to allow others to reproduce the experiment and, of course, should provide clear and sound conclusions.

Given that the sole outcome of the assignment is a report, the quality of the document (i.e., structure of the report, precision of the results reported, quality of writing) is of paramount importance. There is no suggested template for the report structure. The goal is to avoid the rather passive situation in which the students just fill in a given report structure. On the contrary, defining the best structure for the report is part of the goals of the assignment. The teacher is fully available to discuss the proposed structure with the students, as well as any other aspect of the report. Some of the "PL classes" are specifically devoted to provide such support to students. In addition to the report, the students must submit a folder with all the programs and tools used in the work.

3. Resources

Students are supposed to use their own computers.

4. Calendar and miscellaneous

The assignments must be done by groups of up to 3 students according to the following calendar:

- September 27th – Send an email to the teacher (paquete@dei.uc.pt) with the names and emails of the students that constitute each group. Note that the group of students must remain the same for all the assignments.
- October 25th – Submit the assignment report for the first milestone.
- November 22nd – Submit the assignment report for the second milestone.
- December 20th – Submit the assignment report for the third milestone.
- Oral defense from 6 to 9 January (to be defined later on for each group).

Submissions of the reports should be uploaded at Inforestudante. Submissions after the deadline (at 23h59) are not possible.

Plagiarism means mandatory fail in the course and internal (UC) disciplinary procedure. Please, refer adequately all text and material you take from the Internet. All parts of the report must be written by the students and not copied & pasted & changed from the Internet.