Experimental Methods in Computer Science

DEI-FCTUC, 2021/2022

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.). Even for the independent variables, the basic type of experiments only changes one factor at the time (more complex experiments use a factorial approach in which several factors varied together to make the experiment more efficient and allow the study of possible interactions among factors. But, for the time being, we consider only one factor at the time).

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 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 the experiment and execute all the steps to evaluate the following general problem statement:

There are several exams to be scheduled at the Department of Informatics Engineering and there is a number of free time slots. You have done a program that automatically schedules exams, that is, it assigns an exam to a time slot such that every student does not have to attend more than one exam simultaneously and it uses the least number of time slots available. Assume that the time slots do not overlap in time.

Given the success of your program, you are now planning to sell it to other universities. An important aspect to potential clients is to know how much time they need to wait to obtain a schedule with the minimum number of time slots. This should depend on the number of exams, the number of time slots available, and the number of students that have exams. Your goal is to characterize the performance of your program for several scenarios.

Although this is a somewhat artificial problem statement (the main goal is pedagogic and not to represent a realistic research question), it is worth noting that it requires all the steps of a real experiment. Furthermore, the type of measurements required by this experiment (performance) is highly representative of experiments with computers.

The following assumptions and recommendations should be taken into account in the work:

- There are two programs (code1.c and code2.c) that implement two randomized backtracking algorithms to solve the exam scheduling problem, which are

available at Inforestudante. You can decide to use these two implementations or to use others that are publicly available (the source must be referenced in the assignment report), or even to implement your own.

- Both implementations require three arguments: a random seed, the maximum CPU-time in seconds to run the program and the name of the file that contains the input data. Both output the following information for each input data: the minimum number of time slots that is possible to use, or value -1 in case the minimum number of times slots is not found within the maximum time defined by the user. Since some experiments may take too much time, define the maximum CPU-time at your choice.
- The file code1.c implements a backtracking algorithm that will try to find a schedule with n time slots. If it finds one, it will try to find a schedule with $n-1$ time slots, and so on, until it does not find a feasible schedule for some number i of time slots. In that case, the backtracking algorithm terminates and outputs value $i+1$. The file code2.c implements a similar search strategy, but it will first try a schedule with one time slot, and if it does not exist, it will try with two, and so on, until it finds a schedule with j times slots. In that case, it terminates and outputs value j.
- There is an input data generator in python (gen.py) that is also available at Inforestudante. This generator creates randomly generated input data that can be read by the two implementations mentioned above. You need to define the number of exams, the probability (p) that each pair of exams will have a student in common, the random seed, and the name of the file that will store the input data. Each file that is generated contains two integers (n and m) in the first row, and each of the next m lines contains two integers, the indices of two exams for which there exists at least a student in common. The exams are numbered from 1 to n. You may consider other generators with different features.
- You should also set up the number of times the implementation is ran in order to obtain adequate measurements. In case of outliers, you have to decide whether they are relevant or not for the analysis.
- Note that if the number of exams is too small, you may have difficulties in measuring the time accurately, but if the numbers are too large you may have other problems such as the time taken by the experiments or the programs may reach some limits of the machine (e.g., memory) that may complicate the experiments. The implementations can deal with at most 299 exams, but you can define a larger value.
- 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

The outcome of the assignment is a written report (PDF file). The report should describe all the steps of the design of the experiment with enough detail to allow others to reproduce the experiment and, of course, should provide clear conclusions about the problem statement.

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 keep a folder with all the programs and tools used in the work that allow the complete results to be reproduced.

3. Milestones

There are three milestones, each of which covers particular techniques that are discussed throughout the course:

1. *Exploratory Data Analysis and Linear Regression*
2. *Pre-registration of hypotheses*
3. *Hypothesis testing*

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 the first and the third 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 experiments with enough detail to allow others to reproduce the experiments and should provide clear and sound conclusions. The outcome of the second milestone is a single written page (PDF file) for the pre-registration of hypotheses to test in the third milestone. The goals for each milestone will be discussed in the classes.

The quality of the reports (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.

4. Resources

Students are supposed to use their own computers.

5. Calendar and miscellaneous

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

- October 1st – Submit a single PDF file with the names, student IDs and emails of the students (only @student.dei.uc.pt or @uc.pt) that constitute each group. Note that the group of students must remain the same for all the assignments.
- November 9th – Submit the assignment report for the first milestone.
- November 23nd – Submit the assignment report for the second milestone.
- December 20th – Submit the assignment report for the third milestone.
- Oral defense from 5 to 7 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.