TASK I (40 marks)

Brief summary of the task

In this task you will be requested to design, implement, and analyse experiments comparing the performance of different configurations of a nonlinear optimisation algorithm. Some context on the task is provided below, but for the purposes of this task you can think of this problem as just an abstract data-generating process. **Focus on the actual task of designing, running, and analysing the experiment as described in the specific activities**.

Important: You must use the provided R markdown file **Task01.Rmd** to enter your solutions. This is the only file you will submit for this task. If you submit your solution in any other format it will not be marked.

Important: Do not install or load any additional packages when building your solution. All required/allowed packages are listed and loaded in the "Setup" code block at the start of **Task01.Rmd**

Important: For this task, you must submit the following files:

- **Task01.Rmd**, containing your solution to this task.
- **Task01.pdf**, the PDF produced by "knitting" the solutions file Task01.Rmd.

Consistency checking: You can find the checkfile for this task on Blackboard, together with this document, as file *Task01_checkfile.Rmd*. You can perform consistency checking using the **consistency_checker()** function provided by package **SCEMChecker**:

```
# Example:
# If needed, install SCEMChecker
# remotes::install_github("fcampelo/SCEMChecker")
library(SCEMChecker)
mycheck <- consistency_checker(template_path ='Task01_checkfile.Rmd',
                               submission_path = 'Task01.Rmd')
summary(mycheck)
```

Task Context

Complex optimisation tasks appear in several different areas of science and technology, such as:

- Optimisation of drug combination and dosage in cancer therapies (Example)
- Financial portfolio optimisation under uncertainty (Example)
- Design of business processes (Example)
- Topology optimization of static engineering structures (Example).

Population-based metaheuristics (such as Genetic Algorithms) are a common alternative for the solution of these types of optimisation problems. A popular method for continuous optimization is known as *Differential Evolution* (DE)[1,2], which combines a simple algorithm structure with a robust performance for a variety of small- and medium-sized problems. The specific details of this method are not particularly important for this task, but feel free to check the references mentioned in the footnotes in case you are curious about it.

In this task, suppose that you are working as the data scientist in a small company that is investigating different configurations of the DE algorithm for the solution of a certain class of problems that emerge in one of that company's core activities. For that, you will need to design, run and analyse experiments using the knowledge you acquired in this unit. These activities are detailed in the following sections.

Generating data for your experiments

To generate data for the experiments in this task, you will need to use the **development** version of R package ExpDE . This is automatically installed and loaded for you in file **Task01.Rmd**. Code examples of how to generate your experimental data are also provided as pre-populated code blocks in **Task01.Rmd.**

Some code chunks / lines are marked as "DO NOT CHANGE" in **Task01.Rmd**. **You must not change those.** They define important parts of your task, and if changed will make your solution invalid.

Important: do not change anything marked as "DO NOT CHANGE" in **Task01.Rmd**. Modifying those predefined code blocks / lines may result in you getting zero marks on the task.

[1] Storn R, Price K (1997). *Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*. J. Global Optimization 11(4):341–359. https://doi.org/10.1023/A:1008202821328

[2] Campelo F, Botelho M (2016). *Experimental Investigation of Recombination Operators for Differential Evolution*. Proc. Genetic and Evolutionary Computation Conference (GECCO 2016):221-228. https://doi.org/10.1145/2908812.2908852

Activities

Imagine that some of your coworkers in the hypothetical company described in section *Task Context* have checked the technical literature and found a few candidate configurations of the DE algorithm that appear to be effective for solving a core class of problems for your company. As the resident data scientist, you are tasked with finding out whether there's a significant difference in performance between those configurations, which configuration (if any) is the best one and, based on your results, provide a recommendation of which method should be adopted.

Getting your experimental parameters

Most of the required parameters of your task will be generated based on your student number. In the first code block of file Task01.Rmd you will have to enter your student number. It will look like this:

```
MY_STUDENT_ID <- ABCDEFG ## <-- Replace "ABCDEFG" by your student ID number
```

Important: your student number should be a 7-digit **integer.** If your number has a slash in it (e.g., *1234567/1*), you should use only the part before the slash (in this case, **1234567)**. Note that this is **not** the digits in your student username, but your **student number**.

Important: it is essential that you use your own student number. If your solution is submitted using someone else's student number (or any number that is not your own) your whole Task I may be marked as zero.

To obtain the desired statistical parameters for your experiment, you will see a pre-filled code block in Task01.Rmd (named "gen_exp_params1") with a call to the function:

```
exp_pars <- get_exp_params_1(MY_STUDENT_ID)
```

This function will generate individualised parameters for your experiment. The parameters will be echoed when the function is called and also stored as a list in object `exp_pars`. The output will look like the example below:

```
===

Significance level (alpha): alpha = 0.04

Minimally relevant effect size (MRES): d* = 0.25

Desired statistical power for MRES (desired power): (1-beta)* = 0.8

Number of repeated runs for each method on each problem: nruns = 25

===
```

The values you will get from running the code block "gen_exp_params1" in your **Task01.Rmd** will be the ones that you will need to consider when designing and analysing your experiments in this task. These are: (**1**) the desired *significance level* (α); (**2**) the smallest effect size of practical interest (d^*); (**3**) the desired power $\pi^* = (1-\beta)^*$ for differences equal to or greater than d^*; and (**4**) the desired number of repeated runs for each method on each problem, n_{runs}.

To obtain the algorithm specifications that you will need to compare, you will see another pre-filled code block in **Task01.Rmd** with a call to the function:

```
methods <- gen_methods(MY_STUDENT_ID)
```

This function will generate individualised DE configurations for your experiment. The output will look like the example below:

```
===

Your experiment will compare 4 algorithm configurations

Your selected methods are named:

DE.1234567.A

DE.1234567.B

DE.1234567.C

DE.1234567.D

Each method can be passed by name as the argument method.name of function ExpDE2()

If you want to inspect the specificities of each method, check the output list.

===
```

The specific internal details of the methods are not important for this task, and you'll be able to pass just the name of each method directly to the data-generating function. The full configurations are stored in object "methods", in case you want to check them (which is **not** necessary for this task). If at any point you need to retrieve the names of your methods, they will be stored in a variable called `method.names`. You can also re-run `gen_methods(MY_STUDENT_ID)` at any point, and it will also you the method names.

Question 1.1: In this first question, your objective is to compare the mean loss of your methods (the ones generated by the call to `gen_methods()`) on a *single* problem. A sample code that you can use to generate a single observation is provided in **Task01.Rmd**. You will need to adapt it to generate your full experimental data. Based on your experimental parameters (the ones generated automatically in **Task01.Rmd** by the call to `get_exp_params_1()`), you are required to:

a. Formalise the statistical hypotheses that you are testing.

b. Run the experiment defined by your parameters. The test problem will be defined automatically for you in **Task01.Rmd**.

c. Estimate the mean loss and associated confidence interval (at the significance level provided in your experimental parameters) for all methods, and generate a corresponding plot of the CIs.

d. Perform the appropriate parametric statistical test to estimate whether at least one of the methods' mean performances is significantly different from the others, at the significance level provided in your experimental parameters. Assume that the observations obtained for each method represent independent samples.

e. List the main statistical assumptions of the statistical test used and produce residual plots to check if they are satisfied.

f. Based on the results obtained in (d), perform a post-hoc Tukey test to determine which (if any) of the methods have significantly different mean performances from which others.

g. Use your results to recommend one of your methods (the one with the smallest estimated mean loss, regardless of significance) as a "*reference*" method for further experiments.

More detailed instructions of how you should enter your responses to each item are given in **Task01.Rmd**.

Important: Remember that these optimisation methods are trying to *minimise* a loss function, which means that **smaller = better**.

Question 1.2: Based on your preliminary results obtained in Q1.1, your next task is to design and execute a proper experiment to test the mean loss of the candidate methods on the class of problems of interest. For that, you will now run the methods across multiple test problems, to try to discover which method has the best expected performance for other problems belonging to the same class as the test ones. A sample code that you can use to generate a single observation is provided in **Task01.Rmd**. You will need to adapt it to generate your full experimental data. Part of the task of designing this experiment involves determining the effective sample size for this test, i.e., the number of test problems required to achieve the desired statistical properties.

a. Formalise the statistical hypotheses you are testing.

b. Estimate the required sample size n_{probs}. For this sample size estimation:

    - Assume that the value of the minimally relevant effect size d^* that you obtained from `get_exp_params_1()` refers to the smallest difference of practical interest (expressed as the Cohen's d coefficient) between any two methods, paired by problem.

    - Calculate the sample size as you would if you were performing *all-vs-one* paired comparisons of the means, with two-sided alternative hypotheses. Use the Bonferroni correction to adjust the significance level for the number of comparisons.
    (Note: this approach usually slightly overestimates the required number of problems, but it is more intuitive than trying to calculate required sample sizes based on omnibus tests such as anova.)

c. Run the experiment and generate your data.

d. Perform the appropriate parametric statistical test to estimate whether at least one of the methods' mean loss across problems is statistically significantly different from the others, at the significance level provided in your experimental parameters. Assume that the observations obtained for each method on each problem represent independent samples.

e. Based on the results obtained for the previous item, perform a post-hoc *all-vs-one* Dunnett test to determine which (if any) of the methods have significantly different mean loss from the reference one (using the method selected at the end of Q1.1 as the reference).

f. Use your results to recommend one of your methods (the one with the smallest estimated mean loss, *regardless of significance*) as the one you would recommend for your company.