MT 4113: Computing in Statistics, Programming Project 1
Set: Weds, 12th October, 2016
Due: Noon on Friday, 28th October, 2016

1 Your assignment

Design and implement a simulation study to examine the **size** and **power** of **three one-sided** statistical **tests** under a variety of data distributions. The three tests are: (1) two-sample one-sided t-test; (2) one-sided Mann-Whitney test; and (3) a one-sided Monte-Carlo (MC) test of your choice. It's up to you to choose which distribution(s) to simulate from, what sample size(s) of data to simulate each time, how to set up the MC test, and how many simulations to perform. You may want to focus on simulation scenarios you feel will best bring out the differences between the methods, or highlight their relative strengths and weaknesses. Please justify your choices in your report. I provide some background material and some hints and tips below. You must use the software `R` for your simulation study.

1.1 What to hand in

Please pay close attention to this section and hand in the correct files, in the correct formats. If anything is unclear to you, please ask me well in advance.

Before noon (12:00) on the due date, please upload to MMS (under Project 1), archived into a single zip file:

- A short report (Word document, pdf, etc), that explains what you did and what you found. This report should be understandable by anyone who has taken MT4113 - in other words you don't need to explain in any depth how each test works or what size and power are, but you do need to say what you did (and why), what you found and what conclusions you draw from your results. If you tried some things and they did not work out, feel free to include information about this in your report.
- A file (text, Word document, pdf, etc) that gives evidence you spent time designing the simulation program before you started coding. This could be, for example, an outline of the functions that will be written with rough pseudocode inside. You're welcome to comment on your design choices.
- One or more **text files** (with either an .r extension) containing code to run the simulation. I would like to be able to paste the code from this file or these files into `R` get **exactly** the same results that you did.

You do not need to provide evidence that you tested your code, although you'd be wise to test it! If you find it doesn't work as expected, put this in your report (otherwise I'll think that you think it works!). You don't need to put any error checks on the inputs to each function, so long as you document what inputs are required.

If you hand work in late, it will be subject to the late work penalty described at this web site `http://www.st-andrews.ac.uk/maths/current/ug/information/latepenalties/` and summarised as: *A late piece of work is penalised with an initial penalty of 15% of the maximum available mark, and then a further 5% per 8-hour period, of part thereof.* Do not wait until five minutes before the deadline to upload your work to MMS – if the upload fails at that point, your work will be discounted as late.

Because this assignment counts towards your final grade, it is important that you do not collaborate with others in completing the work. *You should be comfortable with the following statement, which you should put as a comment at the beginning of your code file:*

```
#I confirm that the attached is my own work, except where clearly indicated in the text.
```

If you got stuck and needed to ask your peers, please use comments to tell me in your code file where you got help from others. If you are really stuck, please feel free to ask a question on the Help forum of the class Moodle site. My answers there can be seen by everyone, making it fair for all.

For more information, see the "Academic misconduct' section of the university web site `http://www.st-andrews.ac.uk/students/rules/academicpractice/`. This holds for all assessed project work on this course. Plagiarism cannot be tolerated on this or any other assignment for this module.

1.2 Marking scheme

I'll look over everything you hand in, and assign marks out of 50, as follows:

- 10 for scope of the work (how insightful the simulation study you designed should be in highlighting the main features of each test used; how ambitious it is)
- 5 for evidence of good program design (pseudocode, function design, etc)
- 15 for correctness of the code
- 15 for nicely written, well documented, modular code (even if it doesn't work!)
- 5 for quality of the statistical report

2 Background material

2.1 Statistical hypothesis testing, size and power

Hopefully you will remember something about statistical hypothesis testing from previous courses. Take the two-sample t-test as an example. We have two samples of data, $x_1, x_2, \dots, x_n$ and $y_1, y_2, \dots, y_n$, which we assume are a i.i.d. samples from two normal distributions, each with unknown mean and variance. There are two versions of the test: one that assumes the two variances are equal, and one that does not (see any basic stats book for details; this is also mentioned in the supplemental materials for the *Computer-intensive Statistics* part of this module). We wish to test the null hypothesis, H_0, that the samples come from normal distributions that have the same mean, i.e., $H_0 : \mu_x = \mu_y$. If the null hypothesis is false, then some alternative must be true; in this assignment we're working with one-sided tests, so we'll assume an alternative hypothesis $H_1 : \mu_x > \mu_y$. To test the null hypothesis, we construct the test statistic

$$t = \frac{\bar{x} - \bar{y}}{s_{\bar{x}-\bar{y}}} \tag{1}$$

where $\bar{x}$ and $\bar{y}$ are the sample means and s is the standard error of the difference between the means[1]. If the null hypothesis is true, t follows a Student's t-distribution with mean 0, variance 1 and $n_x + n_y - 2$ degrees of freedom (where n_x and n_y are the sizes of the two samples). The larger the observed value of t, the less likely it is that the null hypothesis is true. Given an observed value of t, we can compute the probability that (or frequency with which) a value this large[2], or larger, would be observed if H_0 is true – this computed probability is called the *p-value*. If the observed p-value is equal to or smaller than some pre-defined value, then we judge that it is so unlikely the data could be generated under H_0 that we reject the null hypothesis. The pre-defined value is called the *α-level*, and it's up to us to set the α-level for our test before we see the data. By convention, people usually use $\alpha = 0.05$.

Given this procedure, even if the data really were generated from a two distributions with the same mean (i.e., H_0 was true), we should incorrectly reject H_0 $\alpha \times 100\%$ of the time. Incorrectly rejecting H_0 when it is actually true is referred to as "making a *type 1 error*" (often written type I error).

Sometimes tests don't do what they are supposed to, and incorrectly reject the null too often, or not often enough. This is particularly likely to happen if some of the underlying assumptions are violated (for example, in the case of our t-test, if the data do not come from a normal distribution). It can also happen with small sample sizes for tests that rely on large-sample asymptotics. The *size* of a test is the actual proportion of time that H_0 is incorrectly rejected, given that it is true.

What about if H_0 is actually false and H_1 is true? In this case, we'd be making an error if we failed to reject H_0. Incorrectly failing to reject H_0 when H_0 is actually false is called making a *type 2 error* (often written type II error). The *power* of a test is the proportion of the time we correctly reject H_0 given that H_1 is true.

[1] How this is calculated depends on whether the population variances are assumed to be equal or not, but it's basically a function of the two sample variances (s_x^2 and s_y^2) and the sample sizes (n_x and n_y)

[2] Two sided tests use the absolute value of t, while a one-sided test of $H1 : \mu_x < \mu_y$ would use $-t$

2.2 The tests

You'll find everything you need to know about the two-sample t-test and Mann-Whitney test in any standard stats book. The former tests H_0: $\mu_x = \mu_y$, given two samples of data which we assume are i.i.d. from normal distributions with unknown variance. The latter tests H_0: $M(x) = M(y)$, where M is the median, given two samples of data assumed to be i.i.d. There are actually several variations of the Mann-Whitney test (also called the Wilcoxon rank sum test), and I don't mind which you use.

For the Monte-Carlo test, I'll leave you to choose a suitable one! There are plenty of ideas in the background reading to the *Computer-intensive statistics* part of this module, for example.

3 Hints and tips

- You're welcome to use any built-in packages available for `R` (i.e., packages that load automatically when `R` loads, and do not require a specific `library` or `require` function in your code). I sugest you use the `t.test` and `wilcox.test` funtions (in `R`; a Mann-Whitney test is sometimes called a two-sample Wilcoxon test); you'll need to write your own MC test function.
- If you're struggling to see where to begin, look over the example simulation code in `R` I gave in class (and on the Moodle site, in the Project 1 folder) – this is just an amended version of the simulation we built for Practical 3 to make it easier to run (note how the main code file and the driver file interact).
- The size of a test can only be evaluated by generating data from distributions where H_0 is true. By contrast, power can be evaluated over variety of different values of H_1, ranging from values close to H_0 (i.e., μ_x is only just greater than μ_y), where power should be low (close to the size), to values very far from H_1 (i.e., $\mu_x >> mu_y$), where power should be 1. So, you'll need to think about what values of H_1 seem appropriate. $\mu_x - \mu_y$ in this context is sometimes called the "effect size" or "raw effect size". Size and (particularly) power are also affected by things like the variance of the two populations, the amount of data in the sample and whether the assumptions of the methods are met or not.
- Some general advice about writing statistical reports is here:
  `https://dl.dropbox.com/u/226563/StatisticalReportWriting.pdf`
  Note that I did not write that advice for this module; it was aimed at students on a module we no longer teach. Nevertheless, I hope you find it useful both for MT4113 and for other modules you take.
- Remember that this project is only 20% of your total mark, and that this course is just one of several you're taking. So, don't try to simulate everything! Instead, focus on just one or just a few aspects of the things you think will affect size and power, and investigate those. Don't try to do a fully comprehensive simulation study of the properties of three tests under gazillions of different scenarios! A small amount of well justified and executed code is what I'm looking for, together with a concise and insightful write-up.