**Lab 2: Basic concepts I**

***2.1. The P-value, type-I and type-II errors***

The **P-value** gives the probability to observe your effect size (e.g. the measured difference between sample means), or an even larger effect size, given that the null-hypothesis is true. This is equal to the **type-I error** in your test - the risk that we reject a true H0 (i.e. conclude that there is a difference among groups, even though there really is none). The **type II-error** is instead the risk that we fail to reject a false H0 (i.e. there is in fact a difference between groups, but our data is not accurate/large enough to statistically verify this difference).

You want to test if there are effects of studying statistics on solving non-numeric puzzles. You start with comparing the performance on a specific task for 1st and 4th year students at a statistics program at Uppsala University, with a sample size of 10 students per year-class. What you do not know is that, based on a much larger world-wide study, there actually seems to be no effect of studies in statistics on solving this particular task (i.e. H0: mean 1 = mean 2, is true). Based on that large-scale study, the average score on the test is 30, and the standard deviation is 10.

Assuming that those estimates of the mean and variance are very accurate also for our Uppsala students, we can generate hypothetical statistical samples of Uppsala students, which would simulate the type of data you would have gotten, had you carried out your small study in Uppsala. For this, we can use the **rnorm()** function: *rnorm(number of samples, sample mean, sample standard deviation)* ; see lab 1:

**Year1 <- rnorm(n=10,mean=30,sd=10)**

**Year4 <- rnorm(10,30,10)** #Note that I did not need to type out “n”, “mean” and “sd”

Note also how I have used the **#** symbol, which tells R not to run the code next to it on the same line. This is handy for annotating your code.

You can plot the distribution of the data using different plotting functions in R. One such function is ***hist()***, which plots the density distribution of values (*how common is it that individuals get a particular score on the test*?). If you want the graphs next to each other, try first typing in:

par(mfrow=c(1,2)) # figure panel contains one row, and two columns.

and then run the histograms below. You can also type ***?hist*** to find out more about the ***hist()*** function, and what alternatives there are for plotting with it.

hist(Year1)

hist(Year4)

Looking at the histograms the data hardly look normally distributed – but remember, you sampled them from a normal distribution. The reason the histograms look the way they do is because you have very small samples of only 10 individuals, and by random chance, it is unlikely they will end up following a perfect normal distribution.

A t-test can be performed to assess a statistical difference in means between two groups when the response variable is (presumably) normally distributed and sample size are small, so that seems appropriate to apply in our case. In the next lecture you will learn more about the basic principle behind the t-test and when to use it. For now, let’s use the t-test with the general understanding that it tests for statistically significant differences in means between two groups measured for the same (normally distributed) variable. We can calculate the p-value using:

t.test(Year1, Year4, var.equal=TRUE)

***2.1a. What did you find, and what does the different outputs from the model mean? Did you find the same thing as your class-mates (quantitatively and qualitatively)? Discuss with each other and make sure you understand the general output and result.***

The world-wide large-scale study that found no effect of statistical education on completing the task performed the same experiment (10 1st year vs. 10 4th year students) at other universities; in total 1000 experiments at different places. ***Before running the analyses below, what is your expectation if we perform a test for each experiment separately – that is, 1000 tests – will they all be non-significant?***

OK, so let’s run some code that allows you to sample 10 Year1 and 10 Year4 students in 1000 experiments and run a t-test on each of those experiments:

P.value <- 1:1000 #create a vector where you can save your P-values

for (i in 1:1000) { #use a for-loop to run 1000 experiments

Year1 <- rnorm(10,30,10) #sample size=10, mean=30, sd=10

Year4 <- rnorm(10,30,10) #sample size=10, mean=30, sd=10

P.value[i] <- t.test(Year1,Year4, var.equal=T)$p.value} #Save the P-values

The code above uses a for-loop to re-iterate random sampling for two groups and then calculate t-test statistics on the two samples, 1000 times. You now have the P-values for the 1000 experiments stored in the vector “**P.value**”. Let’s have a look at the distribution of P-values using:

hist(P.value, breaks=100)

You can also calculate how many of the P-values that were significant. For example:

sign.tests <-ifelse(P.value <0.05, 1,0)

This creates the vector **sign.tests** (you can name it something else if you want) with the length of 1000, filled with either ones or zeros, indicating whether or not the test was significant at an alpha level of 0.05. Type:**?ifelse** to find out more about the function we used. We can sum the vector to find out how many of the tests that were significant.

sum(sign.tests)

***2.1b. How many tests were significant? How does it compare to your class-mates’ results? Is it what you expected, and how does it relate to what you learnt about type I error rates in class?***

***2.1c. What would you conclude if you had done a better job of sampling in the experiments?*** A sample size of 10 is by no means very large, and perhaps a larger sample size would cause less false positive tests (type-I errors)? ***Think before you run the test – what do you expect to find?*** Then modify the code above with the for-loop to re-run the 1000 experiments with a sample size of 2000 instead of 10 per year-class. ***Discuss with your class-mates after you have digested the results – was it what you expected?***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You don’t give up, do you? You choose another task, and for this task year4 students typically do better and score 30, while year1 students typically score 29 (the standard deviation is still 10 for both).

***2.1d. Try to rerun your hypothetical experiments and answer the same questions again.*** ***Is there a difference now in how sample size (10 or 2000) affects the number of significant tests? More generally, how are type-I and type-II errors affected by sample size?***

To help you think further about this, it would be nice to easily change sample size, effect size, sampling variance quickly to see how this impacts the t-test. This is an excellent time to show you have you can write ***functions*** in R to speed up tasks like this one (that is- perform computations while changing variables/parameters to see how this affects the output of calculations and simulations in R). I’m here trying to kill two birds with one stone – below is an example how you can prompt Chat-GPT to help you writing and understanding R-code. So, remember, make sure to use Chat-GPT to improve your understanding, but don’t use it in place of thinking for yourself. Have fun!

**My prompt to Chat-GPT:**

*Hi! I want to show students how to write a function wrapped around a for-loop in R. I have a specific task I will provide below. The loop samples two groups and compares them with a two-sample t-test. In the loop you can change: the number of experiments (now set to 1000), the sample size for each group (now 10), the mean (now 30 for both groups) and the sampling variance (now sd = 10 for both groups). I want to write this as a function where it is possible to change all these parameters.*

*The aim is then to store the p-values from the t-test and later plot these so that the students can explore the effect of sample size, effect size, and sampling variance on the statistical power of a test and how type I and type II errors are affected by these changes. It's an assignment for students, so pedagogic explanation of the steps is preferable.*

*Here follows the R code:*

*P.value <- 1:1000 #create a vector where you can save your P-values*

*for (i in 1:1000) { #use a for-loop to run 1000 experiments*

*Year1 <- rnorm(10,30,10) #sample size=10, mean=30, sd=10*

*Year4 <- rnorm(10,30,10) #sample size=10, mean=30, sd=10*

*P.value[i] <- t.test(Year1,Year4, var.equal=T)$p.value} #Save the P-values*

**Output from Chat-GPT:**

# Define a function for the simulation

simulate\_ttest <- function(n\_experiments = 1000, # number of repeated experiments

sample\_size = 10, # sample size per group

mean1 = 30, # mean of group 1

mean2 = 30, # mean of group 2

sd1 = 10, # standard deviation of group 1

sd2 = 10) { # standard deviation of group 2

# Create an empty vector to store the p-values

p.values <- numeric(n\_experiments) # more efficient than 1:n

# Run the experiments

for (i in 1:n\_experiments) {

# Generate random samples for the two groups

group1 <- rnorm(sample\_size, mean = mean1, sd = sd1)

group2 <- rnorm(sample\_size, mean = mean2, sd = sd2)

# Perform two-sample t-test (assuming equal variance)

test\_result <- t.test(group1, group2, var.equal = TRUE)

# Save the p-value from this experiment

p.values[i] <- test\_result$p.value

}

# Return the vector of p-values

return(p.values)

}

### Run simulation with default parameters

pvals <- simulate\_ttest()

# Plot histogram of p-values

hist(pvals, breaks = 20, main = "Distribution of p-values",

xlab = "p-value", col = "skyblue")

# What fraction of tests are significant?

sign.tests <-ifelse(pvals <0.05, 1,0)

sum(sign.tests)/length(pvals)

### Run simulation with other experimental parameters

pvals <- simulate\_ttest(1000, 100, 30, 25, 10, 10) #mean1 = 30 vs mean2 = 25

# Plot histogram of p-values

hist(pvals, breaks = 20, main = "Distribution of p-values",

xlab = "p-value", col = "skyblue")

# What fraction of tests are significant at alpha = 0.05?

sign.tests <-ifelse(pvals <0.05, 1,0)

sum(sign.tests)/length(pvals)

***2.2. Multiple testing***

***The problem with multiple testing***

*• If we make 10 independent tests, each will have a 95% probability to be significant even though the H0 is true (𝛂 = 0.05).*

*• Probability that none of the 10 tests will be significant: 0.95^10 = 0.60*

*• Thus, there is a 40% risk that at least one test will be significant by chance – a* ***type-I error***

*• By correcting 𝛂 to 0.01 the risk of type-I error becomes less than 10% (1-0.99^10) - Great! However, by decreasing 𝛂, we increase the risk of accepting a false H0 (a* ***type-II error!****)*

Continuing with your experiments above, you choose to expand your horizons to testing performance in a wider set of 100 tasks. Let’s generate 100 tests (one for each task) with a sample size of 50 students in each group. And let’s assume that 4-year students score better on the tests on average (2 points better averaged across the 100 tests).

P.value <- 1:100 #save the P-value for each of the 100 tested tasks

for (i in 1:100) { #use a for-loop to run 100 studies, on per task

Year1 <- rnorm(50,30,10) #sample size=50, mean=30, sd=10

Year4 <- rnorm(50,30+rnorm(1,2,1),10)# mean 2 points higher on average

P.value[i] <- t.test(Year1,Year4, var.equal=T)$p.value} #Save the P-values

hist(P.value)

sign.tests <-ifelse(P.value <0.05, 1,0)

sum(sign.tests)

You did find some significant year-differences in the end, for some of the 100 tasks, and you can’t brush that smile off your face. Your friend, however, asks you to correct for multiple testing. Your friend explains that because you have used an alpha level of 0.05, and did run 100 tests, you expect about 5 tests to be significant by chance even without a true effect of statistical studies on performance. To please your friend, let’s try the **p.adjust**()-command to correct the P-values for multiple testing.

Corr.P <- (p.adjust(P.value, "bonferroni"))

hist(Corr.P)

sign.tests <-ifelse(Corr.P <0.05, 1,0)

sum(sign.tests)

***2.2a. What happened?*** The Bonferroni method is the classic, very conservative method - It multiplies all your P-values with the number of tests you made, so it is hard to find significant effects with this method when you do many tests (type II-error rate becomes high). There are other, less conservative methods that may be more reasonable for multiple testing, (type: **?p.adjust**) like the Benjamini & Hochberg method (aka: False Discovery Rate) that is often applied to bioinformatic data:

Corr.P2 <- (p.adjust(P.value, "fdr"))

hist(Corr.P2)

sign.tests <-ifelse(Corr.P2 <0.05, 1,0)

sum(sign.tests)

Multiple testing naturally means that you will discard some results that would have been considered significant on their own, had you not run many tests. In biology you sometimes therefore try to focus the hypothesis and limit the number of tests *a priori*. With omics data this can be hard, and even unwanted, as you beforehand do not know which genes are candidates. Towards the end of this course, you will think more about multiple testing in a more applied situation working on data from a genome-wide association study (GWAS).

***2.3. Biological and technical replication***

There is always variability among statistical samples that comes from both biological sources (individual variation in effects of genes and environment) and measurement errors (related to equipment or the implementation of measurements).

Let’s simulate a micro-array study on gene expression, with focus on expression of a particular candidate gene. You first log-transform your fluorescence intensity scores (measuring the abundance of transcript from your candidate gene) to make them approximately normally distributed, and then zero-center your data so it has a mean = 0 (xi-xbar). We can simulate the variability in this data from two independent sources; the biological source and that from technical measurement error. We use the rnorm-function: *rnorm(number of samples, sample mean, sample standard deviation)* to generate two vectors that describe biological variation and technical variation (measurement error) among 1000 subjects.

Bio <- rnorm(1000, 0, 1)

Tech <- rnorm(1000, 0, 2)

When two sources of error are independent (uncorrelated), they are additive, so the total variance among subjects in your statistical sample is the sum of the two: VTotal = VBio + VTech . Try typing:

Total <- Bio + Tech

You can now check that this additivity of variances is trueby using the **var()** function on each of the three vectors (some small differences are expected due to chance). Remember from the lecture that the variance of a sample (*s2*) is equal to its standard deviation squared.

var(Bio)

var(Tech)

var(Total)

You can also calculate how well you predict the mean expression of the gene by calculating its standard error (SE, or SEM = Standard Error of the Mean), which (recalling from lecture 2) is equal to the standard deviation of the sample, divided by the square-root of the sample size: SE = sd/sqrt(n). A small SE indicates that you are very accurate in your estimate (± 1.96 SE ~ 95% confidence limit, within which the true mean lies with 95% probability).

**2.3a. *Now try to think about what will happen to the prediction of the mean if you:***

1. *double the number of technical replicates (taking two tissue samples from each study subject)*
2. *ii) double the amount of study subjects.*

**2.3b. *What is preferred in this case?* –** think both about the amount of work needed, the costs, and the improvement in the prediction of the mean expression of the gene – discuss with your classmates*.* You’ll find some help by looking at the example in the end of lecture 2.

**2.3c. *ADVANCED* (TOTALLY OPTIONAL) *BONUS QUESTION: After discussing this can you, as a class together, calculate the standard error of the mean expression of the gene (SE) for all three experimental designs*** (***the original, increased biological replication, or increased technical replication).***

This can be done both using R on your simulated datasets (everyone will get similar but slightly different results because you all performed individual simulations of the data), or theoretically based on the assigned standard deviations: sd = 1 for biological variation and sd = 2 for technical error (everyone will get the same answer).

The first hint to solving this rather tricky question is to remember that errors that are independent of each other and contribute to variation in a single factor are additive (in this case, biological and technical variation contribute independently to the expression of the gene). Therefore, these sources of variation can be added up to get the total variation in the gene’s expression (see above). The second hint is to understand that the error in the prediction of the true mean (the SEM) decreases with the square root of the sample size. Hence, to solve this puzzle and calculate the SEM, you must calculate the error in the prediction that stems from technical and biological variation respectively in each experiment, by accounting for the sample size you have for each of the two levels of replication.