Assignment 06

Statistical Computing and Empirical Methods

A word of advice

Think of the SCEM labs like going to the gym: if you pay for gym membership, but instead of working out you use a machine to lift the weights for you, you won't grow any muscle.

ChatGPT, DeepSeek, Claude and other GenAI tools can provide answers to most of the questions below. Before you try that, please consider the following: answering the specific questions below is not the point of this assignment. Instead, the questions are designed to give you the chance to develop a better understanding of estimation concepts and a certain level of **statistical thinking**. These are essential skills for any data scientist, even if they end up using generative AI - to write an effective prompt and to catch the common (often subtle) errors that AI produces when trying to solve anything non-trivial.

A very important part of this learning involves not having the answers ready-made for you but instead taking the time to actually search for the answer, trying something, getting it wrong, and trying again.

So, make the best use of this session. The assignments are not marked, so it is much better to try them yourself even if you get incorrect answers (you'll be able to correct yourself later when you receive feedback) than to submit a perfect, but GPT'd solution.

Introduction

Before starting, make sure you have reviewed the Week 6 lecture slides. Additionally, download the following data files from Blackboard and save them in the same folder as your solutions Rmarkdown. We will use these files for some of the questions in this assignment.

- *session_ab_test.csv*
- *UPMSP_SA_full_results.rds*
- *business_strategies.csv*

This assignment will be done in **RStudio**. To enter your solutions, please use the pre-structured R markdown template provided for this week. This is a similar format to what you will use in the final coursework.

You do not need to submit anything for this assessment. Feedback will be provided on the labs, and in the form of model answers in the following week.

If you want to check that your answers are consistent with what is expected (i.e., that all code blocks are generating the required variables, of the correct types etc.) please use the consistency_checker.R script available on Blackboard (don't forget to set the assignment number to 6). Alternatively, you can simply run the code below in your R console:

```r
install.packages("remotes", repos = "https://cloud.r-project.org")
remotes::install_github("fcampelo/SCEMChecker", dependencies = c("Imports"),
                        force = TRUE)

# Path to your submission file (change as needed)
submission_path <- "W06_Assignment_AnswerTemplate"

# 'mycheck' will contain a data frame with the details of the check
mycheck <- SCEMChecker::consistency_checker(submission_path,
                                            template_number = 6)

summary(mycheck)
```

This assignment relates to this week's lecture on statistical inference for multiple samples, and extends the concepts covered in the lecture to include:

- power/sample size calculations for the two-samples t test
- Group segmentation in A/B testing
- Paired t-tests (when samples are matched)
- Post-ANOVA tests to pinpoint specific differences
- Randomised complete block designs for comparing the means based on matched samples.

Review of Welch's t test

In this first part, we'll review Welch's t-test for the comparison of means of two independent populations, without the need to consider equal variances.

Welch's t-test is implemented using the same function as the one-sample t-test that we learned last week, `t.test()`. There are a couple of different notations that can be used to run it.

To illustrate how to run a Welch t-test in R, we will use data from the USA's National Health and Nutrition Examination Surveys (NHANES) from 2009-2012. This data set is available in R package `NHANES`, and you can access it using:

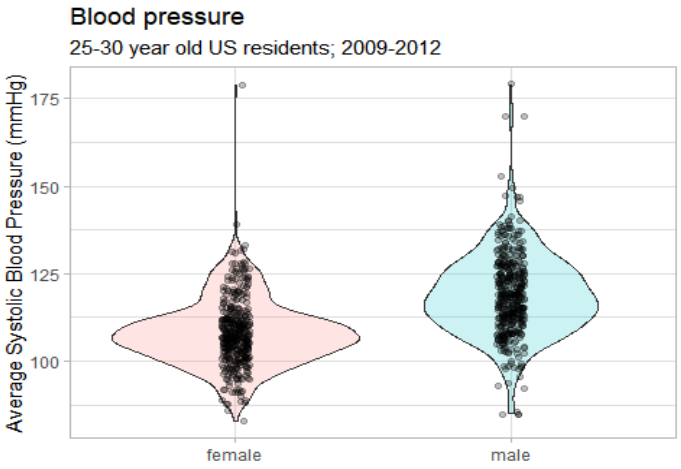
```r
library(NHANES)

## Warning: package 'NHANES' was built under R version 4.4.3

data("NHANES")
```

This loads a dataframe called `NHANES` into your R environment. If you're curious, you can read the complete data description by typing `?NHANES` in your R console.

Q1: To warm up:

a. We'll use the NHANES dataset to compare the mean combined systolic blood pressure reading of males and females in the 25-30 year-old demographic. Load the data using the code provided above, then use dplyr (a combination of `filter()` and `select()`) to create a simplified data frame called `bp.df`, containing only the observations for individuals between 25 and 30 years old (variable `Age`), and only the columns `Gender` and `BPSysAve`. Remove any observations with missing values (i.e., `NA`) in the `BPSysAve` column. Your resulting dataframe should have dimension 825 x 2.

b. Use `ggplot2` to generate a plot of systolic blood pressure vs. gender. Overlay a jittered scatterplot (using `geom_jitter()` and adjusting the `width` parameter) with a violin plot (using `geom_violin()`) to create a richer visualisation. You can also play with ggplot themes (e.g., `theme_minimal()` or `theme_light()`), labelling functions (`xlab()`, `ylab()`, `ggtitle()`) or geom transparency (`alpha = ...`) to create a more visually appealing plot - check the figures in Lecture 06 for code examples. Your visualisation may look like this:



The data visualisation above suggests some differences - and you can use the two-sample t-test to test it. There are two main notations to run this test: vectors-based or formula-based. The second is usually easier when your data is contained in a data frame. Assume that we want to test this at the 99% confidence level:

```
# method 1: vectors
bp.m <- bp.df$BPSysAve[bp.df$Gender == "male"]
bp.f <- bp.df$BPSysAve[bp.df$Gender == "female"]
t.test(bp.f, bp.m, conf.level = 0.99)

##
##  Welch Two Sample t-test
##
## data:  bp.f and bp.m
## t = -14.312, df = 811.52, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -12.715656  -8.828874
## sample estimates:
## mean of x mean of y
##  108.0376  118.8099

# method 2: formula
t.test(BPSysAve ~ Gender, data = bp.df, conf.level = 0.99)

##
##  Welch Two Sample t-test
##
## data:  BPSysAve by Gender
## t = -14.312, df = 811.52, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group female and group male is not
equal to 0
## 99 percent confidence interval:
##  -12.715656  -8.828874
## sample estimates:
## mean in group female   mean in group male
##             108.0376             118.8099
```

Notice that the test provides a similar summary to the one you encountered for the one-sample t-test. You have the value of the t-statistic, the test degrees-of-freedom, the p-value, a statement of the alternative hypothesis, a confidence interval (which in this case is the confidence interval on **the difference of the means**) and the sample estimates of each group. The CI for the difference of two means is calculated using the same template as the CI on a single mean,

$$CI = point\ estimate \pm \text{t-quantile} * standarderror$$

For the difference of two means,

- Point estimate = mean(group 1) - mean(group 2)
- t-quantile = $(1 - \alpha/2)$-quantile of the t-distribution with the Welch-adjusted degrees-of-freedom.
- standard error = SE of the sampling distribution of the difference of two means, calculated as $\sqrt{s_1^2/n_1 + s_2^2/n_2}$.

The easiest way of getting this CI is by running the t-test and extracting it directly from the resulting object,

```
tmp <- t.test(BPSysAve ~ Gender, data = bp.df, conf.level = 0.99)
tmp$conf.int

## [1] -12.715656  -8.828874
## attr(,"conf.level")
## [1] 0.99
```

From this example, you can conclude a few things:

(i) The difference in mean systolic blood pressure between males and females in the 25-30 year old demographic is statistically significant at the 99% confidence level (since $p < 0.01$).

(ii) The *magnitude of the difference* can be estimated as:

```
# CI centerpoint = point estimate of the difference
## (since CI on the mean is symmetric)
cat("Difference of means (Females - Males)")

## Difference of means (Females - Males)

mean(tmp$conf.int)

## [1] -10.77227
```

that is, in this demographic, females tend to have a lower systolic BP than males, by about -10.8mmHg with a 99% CI of (-12.7, -8.83)mmHg, which we can also express as $\Delta\mu_{(F-M)} = (\text{-}10.8 \pm 1.94)$mmHg

c. Now do it yourself. You will test the common idea that marriage tends to have a protective effect on the cardiovascular health of men. For that, nstantiate a new dataset from the NHANES data, containing the following subpopulation:

- Gender: male
- Age: between 30 and 40 years
- MaritalStatus: either “Married” or “NeverMarried” (use `MaritalStatus %in% c("Married", "NeverMarried")` in your filtering function)
- BPSysAve: not missing

Make sure the new dataset only contains columns “MaritalStatus” and “BPSysAve”, and name it `bp.marital`. Then run a Welch t-test to check if the mean systolic blood pressure is significantly different between the two groups, at the 95% confidence level. Store the p-value as variable `pval` and the confidence interval as variable `ci95`. Is the difference statistically significant?

To reflect: what is the population for which this inference applies?

A/B test subgroup segmentation, statistical power, and multiple comparisons

In many digital experiments (A/B tests), not all users respond to a change in the same way. Some subgroups, e.g., different age groups, genders, or devices, may show stronger or weaker effects. **Subgroup segmentation** helps uncover these heterogeneous treatment effects.

When analysing segmented A/B tests, we are essentially performing **conditional comparisons**, examining whether the level effect differs within specific categories of another variable. For example: - Does a new app design improve engagement equally for younger and older users? - Do customers in different regions respond differently to a pricing change?

However, testing across subgroups introduces two main challenges:

1. **Sample size and power**: Splitting data into subgroups reduces the number of observations per test, which reduces power. Each subgroup analysis must have enough observations to detect a meaningful effect.

2. **Multiple comparisons**: If we run many subgroup tests, we increase the chance of false positives (Type I error). This usually requires a **correction for multiple hypothesis testing** to ensure that the probability of false positives remains controlled at the desired level. Another strategy is to use hierarchical or factorial models, but this will not be discussed today.

Power and sample size

The first issue (power / sample size) is something that we need to consider when designing or planning data collection, regardless of subgroup stratification. Although the mathematical details of sample size calculation are more complex than we have time to cover in this course, the basic rationale is generally simple.

- Define the type of H_1 and the significance level α
- Define a *minimally relevant effect size* δ^* (the smallest deviation from the null hypothesis that is of practical interest).
- Define the minimal desired power (π^*) to detect effects equal to or larger than δ^*.
- Obtain an initial estimate of the variance of each group (e.g., from preliminary data collection, practical experience, or theoretical expectations), s_i^2.

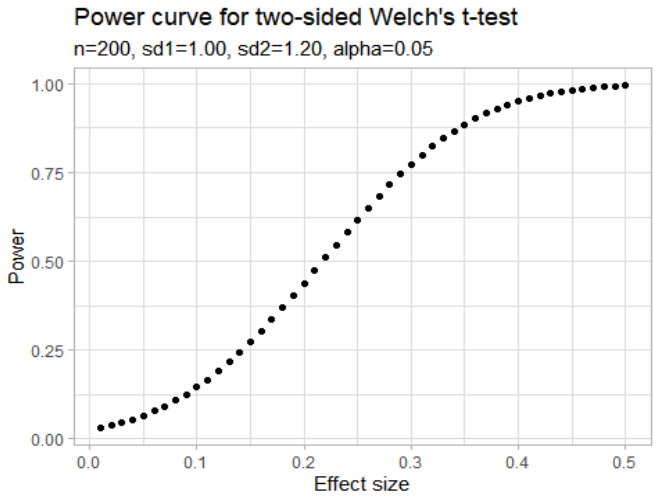
Given these definitions, the necessary number of observations can be easily calculated for the equal sample sizes case (it is a bit more complex for heterogeneous sample sizes, but it can be solved using simulation). See the example below for a hypothetical case:

```r
# install.packages("MKpower") ## <-- install if needed
library(MKpower)

h1    <- "two.sided" # direction of alternative hypothesis
alpha <- 0.05        # significance level
delta <- 0.25        # MRES, minimally relevant effect size
sd1   <- 1           # estimated standard deviation of population 1
sd2   <- 1.2         # estimated standard deviation of population 2
power <- 0.8         # desired power to detect effects >= MRES

power.welch.t.test(delta = delta, sd1 = sd1, sd2 = sd2, sig.level = alpha,
                   power = power, alternative = h1)
```

```
##
##      Two-sample Welch t test power calculation
##
##               n = 307.4149
##           delta = 0.25
##             sd1 = 1
##             sd2 = 1.2
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Note that, if you have a predefined sample size (e.g., if you are doing a retrospective analysis) you pass `n` to the function and leave `power` undefined, and the function will give you the achievable power given the other characteristics. Or you can extract a *power curve*,

```r
n <- 200 # assuming existing samples of 200 observations per group

power.df <- data.frame(delta = seq(from = 0.01, to = .5, by = 0.01),
                       power = NA)

power.df$power <- sapply(power.df$delta,
                         \(x){
                           power.welch.t.test(n = n, delta = x,
                                              sd1 = sd1, sd2 = sd2,
                                              sig.level = alpha,
                                              alternative = h1)$power
                         })
ggplot(power.df, aes(x = delta, y = power)) +
  geom_point() +
  theme_light() +
  xlab("Effect size") + ylab("Power") +
  ggtitle("Power curve for two-sided Welch's t-test",
          sprintf("n=%d, sd1=%2.2f, sd2=%2.2f, alpha=%2.2f", n, sd1, sd2, alpha))
```



Some important notes:

- If you can assume equal variances, a simpler function to use is `power.t.test()`.
- If you're running sample size / power calculations for subgroups in an analysis, make sure to use the standard deviation estimates for those subgroups (which may or may not be the same as the overall population)

Multiple comparisons

Whenever possible, the planning of which comparisons will be performed on a given set of data should be done *before* any exploratory data analysis is performed. Testing hypotheses on the same data used to define those hypotheses is sometimes known as HARKing) (*Hypothesising After the Results are Known*), and it is a common entry point for researcher biases into the analysis. In a very practical sense, it is the same as training and testing a model on the same set of data, and it can lead to an arbitrary inflation of the actual type-I errors (regardless of what the α value is).

Multiple hypothesis testing (MHT) often emerges in practice. One common example is in A/B testing, where the data scientist may be interested not only on the *global* effects of a certain change, but also on *subpopulation* effects (e.g., what is the difference in user engagement between two designs for different age groups?).

One important aspect of testing multiple hypotheses is keeping the overall type-I error controlled. As each test has a probability of type-I error given by α, if you test multiple times, your overall probability of false positives increase. You can see that in the simulation below:

```r
set.seed(2025)

# Parameters
K           <- 10   # number of tests (e.g., 10 subgroups)
n_per_group <- 20   # sample size per group
alpha       <- 0.05 # nominal significance level
nruns       <- 5000 # number of simulation runs

# Function to simulate one experiment with multiple tests
# In all cases, we enforce the null hypothesis as true
# using groups drawn from the same distribution
simulate_experiment <- function(K, n_per_group, alpha) {
  pvals <- replicate(K, {
    x <- rnorm(n_per_group)
    y <- rnorm(n_per_group)
    t.test(x, y)$p.value
  })

  # Did any of the tests (incorrectly) reject the null?
  return(any(pvals < alpha))
}

# Run simulation
results <- replicate(nruns, simulate_experiment(K, n_per_group, alpha))
empirical_error_rate <- mean(results)

cat("Nominal alpha =", alpha,
    "\nEmpirical 'family-wise error rate' =",
    signif(empirical_error_rate, 3))

## Nominal alpha = 0.05
## Empirical 'family-wise error rate' = 0.403
```

Feel free to copy the code below and check the effect of number of tests `K` (or any other parameter of this simulation) on the family-wise error rate (FWER).

There are a number of ways of adjusting the value of α in the pairwise comparisons in order to maintain the FWER at a controlled level. IF the number of comparisons is relatively small, two of the most common ones are called the *Bonferroni* and the *Holm* corrections.

Assuming K planned comparisons, the Bonferroni correction consists of simply multiplying the p-values by K (or, equivalently, testing all hypotheses at a corrected significance level $\alpha' = \alpha/K$). This approach is quite simple, but it results in very conservative tests (i.e., it sacrifices statistical power for a strict control of the FWER).

The Holm correction is a step-wise procedure that is considerably less conservative than the Bonferroni one, while still controlling the FWER. First, the p-values for all tests are computed. The hypotheses and their corresponding p-values are then ranked in *increasing* order of p-values, such that $p_1 \leq p_2 \leq \ldots \leq p_K$. The p-values are then corrected based on their rankings, $p' = (N - r + 1)p$, where r is the ranking of an ordered p-value.

Regardless of the method, R function `p.adjust()` provides an easy way to adjust p-values of multiple comparisons, which you can then compare against the normal (unmodified) α to decide whether or not to reject each H_0. You don't need to sort the p-values before passing them to the

```r
set.seed(2025)

## Update the experimental simulation function to
## include MHT corrections
simulate_experiment_mht <- function(K, n_per_group, alpha, correction) {
  # Generate a vector of K p-values (one for each hypothesis)
  pvals <- replicate(K, {
    x <- rnorm(n_per_group)
    y <- rnorm(n_per_group)
    t.test(x, y)$p.value
  })

  # Vector of corrected p-values
  pvals.corrected <- p.adjust(pvals, correction)

  # Did any of the tests (incorrectly) reject the null?
  return(any(pvals.corrected < alpha))
}

# Run simulations
FWER.df <- data.frame(correction = c("none", "bonferroni", "holm"),
                      nominal.alpha = 0.05,
                      FWER       = NA)
for (i in seq_along(FWER.df$correction)){
  res <- replicate(nruns,
                   simulate_experiment_mht(K, n_per_group, FWER.df$nominal.alpha[i],
                                           FWER.df$correction[i]))
  FWER.df$FWER[i] <- mean(res)
}

FWER.df

##   correction nominal.alpha   FWER
## 1       none          0.05 0.4030
## 2 bonferroni          0.05 0.0446
## 3       holm          0.05 0.0466
```

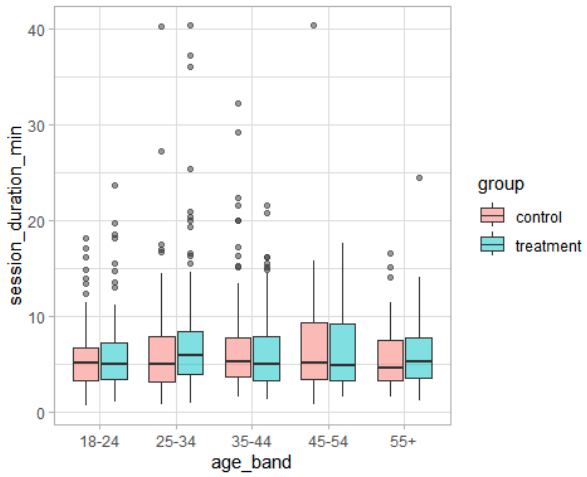
As you can see, both correction methods kept the error rates controlled below the nominal value. In general, Holm's correction is recommended over Bonferroni's, as it results in better statistical power.

Q2. A/B tests - power calculations and multiple hypotheses testing

In this question we'll explore power and sample size, as well as multiple hypothesis testing, in the context of A/B testing. For that, we'll use a synthetic dataset available in file "session_ab_test.csv".

This data contains information about the average session duration of users exposed to two versions of a given web service: a current one (identified as "control") and a new one (identified as "treatment"). The data also contains information about the age band of each user. In this experiment, you want to explore whether the new version ("treatment") improves (=increases) the mean session duration in relation to the current one, across different age bands.

a. Use `read.csv()` to load the dataset "*session_ab_test.csv*" into a variable called `df.ab1`. Use ggplot2 to produce a figure showing boxplots for each web service version, on each age band. For this you'll need (at least) `geom_boxplot()`. If you map the aesthetic `fill` (i.e., if you do `aes(..., fill = name_of_variable)`), ggplot2 already sorts out the splitting of data based on each level of the variable. Your plot should look like the one below:



b. The samples seem to be somewhat heavy on the right tail (i.e., with an over-representation of extreme values greater than the mean). As discussed in the lecture, in these situations, when the observations are strictly positive (as is usual with "duration" variables), a common strategy is to work with the log of the response variable. Use `mutate()` to create a new variable in your dataframe called `duration.log`, containing the base-10 logarithm of the session duration values. Store this new dataframe as `df.ab1.log`, and repeat the plot of **Q2a** using the new variable `duration.log` instead of `session_duration_min`. Observe if the boxplots seem to indicate a more symmetric distribution (feel free to replace the boxplots by violin plots if you'd like to check the shape of the density estimates).

You want to compare the mean session duration of the two versions, for each age band. As you can see from your plots, this means performing 5 hypotheses tests, one for each age band (18-24, 25-34, etc.). For this question, imagine that you are interested in detecting if the new version ("treatment") improves the mean session duration by at least 20% over the current one, with a 95% joint confidence level (i.e., you want to control the FWER at 0.05) and a power of 80%.

Conveniently, working with proportional/percent differences is much easier when working in the log-values scale (which is multiplicative) than the raw scale (which is additive). For instance, to detect a difference of

20%, it means we want to be able to detect if $\mu_2 \geq 1.2\mu_1$. Taking logs on both sides, it means we want to detect if $log_{10}(\mu_2) \geq log_{10}(1.2\mu_1) \rightarrow log_{10}(\mu_2) \geq log_{10}(1.2) + \log_{10}(\mu_1)$, which means that the minimally relevant effect size in the log-scale is $\delta^*_{log} = \log_{10}(1.2) = 0.08$.

Assume that, from past experience, you have a reasonably good upper estimate of the standard deviation of the log-transformed session durations, equal to $\sigma_{ref} = 0.3$.

The available information for the next item can therefore be summarised as:

- Directionality of H_1: one sided (i.e., $H_1: \mu_2 > \mu_1$) (TO THINK: why is H_1 not defined as $\mu_2 \geq 1.2\mu_1$?)
- Significance: $\alpha = 0.05$
- MRES: $\delta^*_{log} = 0.08$
- Desired power to detect the MRES: $\pi^* = 0.8$
- Reference value for the standard deviation: $\sigma_{ref} = 0.3$

c. Based on the information provided above, you now have to calculate the required sample sizes for your tests. For this planning stage, assume the *Bonferroni* correction (i.e., that each test would be run at corrected significance level $\alpha/5$). Since we're assuming equal variances, use function `power.t.test()` to calculate the required sample size for each group. Store the calculated `n` (rounded up) as variable `required_n_per_group`.

d. What is the *available* sample size of each `group` under each `age_band`? Use the base R function `table()` to print a table of the available observations for each (`group`, `age_band`) combination (check the examples in the documentation of `table()` to see how). Is your available sample size close enough to your target one calculated in the previous item?

e. From the result of last question, you are now aware that your study is *underpowered* - i.e., it will not have the desired statistical power for your MRES. Instead of giving up, you decide to aggregate your age bands into two broader groups, to get the required sample size: "younger", for users 34 years old or younger; and "older", for users 35 and older. Use `mutation()` to create a new column `new_age_band` in your `df.ab1.log` dataset, with value "younger" if `age_band` is either "18-24" or "25-24", and "older" otherwise. Store this new dataframe as `df.ab1.log2`, then use `table()` to print the sample sizes under the new age bands. Notice that you are now looking at running only 2 comparisons (instead of the original 5) which will also increase your statistical power.

f. Now, use `t.test()` to run the hypotheses tests comparing the mean of the "treatment" version (μ_2) versus the "control" version (μ_1) for each of your two new age bands.

- Make sure that, for each test, you are only using the observations of the relevant new age band. It may be easier to create two temporary datasets, one for each new age band.
- Don't forget to use the directional H_1 defined earlier ($H_1: \mu_2 > \mu_1$)
- Don't forget that you're running the comparisons using the log-transformed values, not the original ones.

Store the p-values obtained by both tests as a numeric vector `pvals.raw`. Then use `p.adjust()` to perform the *Holm* correction on the raw p-values, and store the corrected p-values as vector `pvals.holm`. Inspect the p-values vector - is any of the differences statistically significant at the 95% confidence level?

Paired t-tests

Sometimes you have comparisons in which the assumption of between-samples independence breaks due to the very nature of the question being asked. For instance, if you're testing a before-after effect (e.g., improvement on the condition of patients after the introduction of a new treatment) there is a clear dependency structure in the data: observations coming from the same patient are correlated.

Similarly, when comparing the performance of two classifiers across multiple data sets, a dependency structure emerges due to the difficulty, noise level, and class imbalance of each dataset affecting the performance of both classifiers.

Recall that one way to model the data when we discuss the t-test for two independent samples is:

$$y_{ij} = \mu_i + \epsilon_{ij}, \begin{cases} i \in \{1,2\} \\ j \in \{1,2,\dots,n_i\} \end{cases}$$

In the a *matched-samples* (a.k.a. *paired samples*) scenario, we need to incorporate a component to this model to account for the effect of each *experimental unit* (such as a patient, a test problem, etc):

$$y_{ij} = \mu_i + \beta_j + \epsilon_{ij}, \begin{cases} i \in \{1,2\} \\ j \in \{1,2,\dots,N\} \end{cases}$$

in which β_j represents the additive effect of the j-th experimental unit. Note that in this case the sample sizes for both groups are rigorously the same (since the samples are present in pairs, one for each experimental unit). In this model, μ_i represents the mean of the variable of interest at the i^{th} level of the experimental factor of interest, *after accounting for the effect of the experimental units*; and ϵ_{ij} is the residual term.

If we take the difference of the paired samples, we can remove the effect of experimental units, which focuses the analysis directly on the factor of interest:

$$\begin{aligned} d_j &= y_{1j} - y_{2j} \\ &= (\mu_1 + \beta_j + \epsilon_{1j}) - (\mu_2 + \beta_j + \epsilon_{2j}) \\ &= (\mu_1 - \mu_2) + (\epsilon_{1j} - \epsilon_{2j}) \\ &= \mu_D + \epsilon_j \end{aligned}$$

This means that, if we focus on the paired differences, $d_j = y_{1j} - y_{2j}$, we can easily test hypotheses about differences in the means of interest, $\mu_D = \mu_1 - \mu_2$, by running a simple, one-sample sample t-test on the sample of paired differences for the hypotheses:

$$\begin{cases} H_0: \mu_D = 0 \\ H_1: \mu_D \neq 0 \end{cases}$$

(or using any other value for μ_D under the null hypothesis, or a direction H_1, as needed). This removes the *between-units variability* from the test calculations and increases (often substantially) the statistical power of the test.

Notice that, whenever you have this sort of paired structure in the data, the relevant sample size is the number of experimental units (e.g., the number of patients in a before-after study, or the number of available datasets in the comparison of two classifiers). The number of repeated measures within an experimental unit (e.g., repeated runs of a method to estimate the mean performance) provides a more precise measurement for that unit, but should not enter into the test calculations. In those cases, these repeated measurements need to be aggregated (often as the mean or median) so that, for each experimental unit, there is only one observation of each group; only then can the paired t-test be run with the correct degrees-of-freedom (which equals $N - 1$, with N being the number of experimental units).

Q3: paired t-tests

In the paper *Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances* (LINK), the authors present an advanced method to estimate the required sample sizes when comparing multiple stochastic optimisation algorithms. One of their case studies involves the estimation of the performance of scheduling algorithms on a set of challenging problems, for which the objective was the *minimisation* of a time quantity called the *Makespan*. The experimental data for that paper is available in file "UPMSP_SA_full_results.rds". This dataset contains the results of running 22 algorithm variations (a reference version called "Full" plus 21 simplified versions derived as part of an ablation study) on 57 problem instances from a problem class of interest. Each algorithm-instance pair has a different number of repeated runs.

In this question, you will compare two methods: the baseline one ("Full") versus a simplified version called "No-2SH". The underlying technical question that you will test can be stated as: "*Is the mean performance of the No-2SH method worse (i.e., larger) on the problem class of interest than the reference method?*".

a. Load the full dataset from file *UPMSP_SA_full_results.rds*. Use a combination of `select()`, `group_by()`, `summarise()` and `arrange()` to create a dataset `df.paired.t` with the following characteristics:

- Columns `Algorithm`, `Instance` and `MeanMS`
- Column `MeanMS` should contain the mean `Makespan` value for each algorithm-instance pair
- Only the observations related to algorithms "Full" and "no-2SH" should be present.
- The rows are ordered by `Algorithm`, and within each algorithm they are ordered by `Instance` (note: you can pass multiple ordering variables as parameters to `arrange()`).

For reference, `dim(df.paired.t)` should show a 114 x 3 dataframe.

b. Formalise the hypotheses being tested, in terms of the mean of paired differences, μ_D. (Note the directionality of the alternative hypothesis).

c. Perform the paired t-test to check if the mean of the paired differences of `MeanMS` between the "no-2SH" and the "Full" versions indicates a loss of performance of the "no-2SH" version (i.e., a larger mean). Check the example code above. (Note that you can run this test either by calculating the paired differences and applying the one-sample t-test, or by applying the 2-sample t-test with option `paired = TRUE`). Store the p-value as variable `pval.paired`, the estimated mean of the paired differences as `diff.paired`, and the 95% confidence interval on the mean of paired differences as variable `ci.paired` (all those values can be extracted directly from the object returned by the t-test).

Are the results statistically significant at the 95% confidence level? What was the estimated effect size?

d. Now try running the t-test without pairing, i.e., as a simple 2-samples Welch t-test. Check the p-value: was the test able to detect this difference? (TO THINK: why not?)

Analysis of Variance

In this week's lecture we also saw how we can deploy a method called the *analysis of variance* (ANOVA) to test for differences of means of multiple independent populations. In a sense, you can consider the one-way ANOVA as a generalisation of the 2-samples t-test for more than two means, with the additional assumption of *homoscedasticity* (equality of populational variances). ANOVA tests the hypothesis of the existence of *some* difference in the means (i.e., that at least one of the population means differs from the others) but it does not specify which one. To recall, the statistical model of the ANOVA is

$$y_{ij} = \mu_i + \epsilon_{ij} = \mu + \tau_i + \epsilon_{ij}, \begin{cases} i = 1, \ldots, a \\ j = 1, \ldots, n \end{cases}$$

where y_{ij} is the j-th observation in the i-th factor level; μ_i is the mean of the population of the i-th level, which can be decomposed into a grand mean μ and a level effect τ_i; and ϵ_{ij} is the residual (unmodelled sources of variability) relative to observation y_{ij}. The hypotheses tested by ANOVA can be expressed as either

$$\begin{cases} H_0: \mu_i = \mu_j, \forall i, j \\ H_1: \exists \mu_i \neq \mu_j, \ for\ any\ i, j \end{cases}$$

or, equivalently,

$$\begin{cases} H_0: \tau_i = 0, \forall i \\ H_1: \exists \tau_i \neq 0, \ for\ any\ i \end{cases}$$

As we saw in the lecture, we can run a one-way ANOVA in R using function `aov()`, and examine the results using function `summary()` on the output returned by `aov()`.

Q4: One-way ANOVA

A business growth team tests six acquisition strategies to see which (if any) delivers the highest 30-day revenue per new customer. This KPI is continuous and noisy since customer value varies, but the company wants to know whether mean 30-day revenue differs between strategies, so they run a randomised pilot experiment to obtain some data to verify this.

Your goal as a data scientist is to test whether at least one strategy yields a different mean revenue. If ANOVA is significant, we'll see later how to pinpoint *which* strategies differ from which others.

a. Load the dataset file *business_strategies.csv* into a variable called `df.strat`. Check the variable names and the dataset size. Use a combination of dplyr's `group_by()` and `summarise()`, and summarisation functions `mean()`, `sd()` and `n()`, to generate a summary of the data containing, for each level of factor `strategy`, the sample mean, standard deviation and sample size. Store this into a variable `df.strat.stats`, with columns `strategy`, `meanRev`, `sdRev` and `N`.

b. Use a combination of `ggplot()`, `geom_qq()`, `geom_qq_line()` and `facet_wrap()` to generate a faceted plot containing the normal QQ-plots for each strategy factor level, to check the approximate normality of your data (also, feel free to change the theme, x- and y- labels, etc). Note that `geom_qq` requires the variable of interest to be passed as aesthetic `sample`.

c. Use function `fligner.test()` to run the Fligner-Kileen test of equality of multiple variances (check the examples in `?fligner.test` for reference). Compare the p-value against a significance threshold of 0.05. If $p > 0.05$, it means that there's no strong evidence that the variances are different.

d. Perform the ANOVA on this data to test for differences in the mean 30-day revenue of the different business strategies. Store the object returned by `aov()` as variable `aov.strat`. Print the summary to the console. Assuming you wanted to test your hypotheses at the 99% confidence level, did the test indicate the existence of *statistically significant* differences?

We already pre-checked the assumptions of normality and homogeneity of variances earlier, but if you'd like you can create the diagnostic plots for your ANOVA model using the code below (go ahead and test it).

```r
# Turn the base plot device into a 2x2 grid
par(mfrow = c(2, 2))

# las=1 (optional) makes the text in the y-axis horizontal.
# pch=20 (optional) regulates the shape of the points
plot(aov.strat, las = 1, pch = 20)

# Turn the base plot back into a single panel
par(mfrow = c(1, 1))
```

Post-hoc tests:

When the result of ANOVA is statistically significant, the natural next question is to check **which** means are different from which others. (TO THINK: this is not necessary if the ANOVA fails to reject H_0. Why not?)

Like in the earlier discussion about multiple comparisons, the planning of which comparisons will be done after ANOVA should *ideally* be defined before the tests are run, to minimise data dredging / researcher bias from creeping into the analysis.

The planning of multiple post-ANOVA comparisons must be guided by the technical question underlying the experiment. Whenever possible, the researcher should opt to perform the smallest number of comparisons needed to adequately answer their question, as more comparisons result in lower statistical power due to the required significance corrections.

The most usual post-ANOVA questions involve:

- How do all levels compare to each other (*all vs. all*)?
- How does one level compare to the others (*all vs. one*)?

All vs. all comparisons are common whenever we the data scientist is simply interested in detecting which levels are significantly different from which others, without any prior information or special interest in one specific level. In these cases, the number of comparisons is $K = a(a-1)/2$, where a is the number of levels.

To perform post-ANOVA *all vs. all* comparisons, a common approach is to use a method called Tukey's Honest Significant Difference (HSD). When the ANOVA assumptions are satisfied, this method provides better power than performing multiple t-tests with adjusted values of α, because it uses a pooled variance estimator provided by the ANOVA object.

All vs. one comparisons usually arise when there's a clear reference (e.g., an established gold-standard method that we want to check alternatives against, a proposed method we want to benchmark against others, comparisons of treatment alternatives against a placebo, etc.) In these cases, the total number of comparisons is $K = a - 1$.

For *al vs. one* comparisons, a test that is usually employed for its superior sensitivity in relation to Tukey's HSD is called Dunnett's test.

Please note that it is also perfectly valid to simply run multiple t-tests and correct the p-values using, e.g., Holm's correction, However, Tukey's and Dunnett's tests are preferable as post-ANOVA tests because they use the ANOVA MS_E as a pooled variance estimate, which provides better statistical power when the ANOVA assumptions are reasonably satisfied.

R package `multcomp` provides convenient methods for post-ANOVA tests. An important point is that this package expects the experimental factor used for the ANOVA to be coded as a `factor` variable (instead of a simple `character` one), but this is very easy to do (see below)

Given an `aov` object, you can run a Tukey test using:

```r
library(multcomp)

# Some simulated data
tmp <- data.frame(group = rep(LETTERS[1:4], each = 10),
                  value = unlist(lapply(c(2,4,6,8),
                                        \(x) rnorm(n = 10, mean = x))))

# Make the variable 'group' into a factor-type variable
tmp$group <- as.factor(tmp$group)

# Run ANOVA
myAOV <- aov(value ~ group, data = tmp)
summary(myAOV)

##             Df Sum Sq Mean Sq F value   Pr(>F)
## group        3 171.37   57.12    83.8 2.65e-16 ***
## Residuals   36  24.54    0.68
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

### Tukey test
myTukey <- glht(myAOV, linfct = mcp(group = "Tukey"))
summary(myTukey)

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = value ~ group, data = tmp)
##
## Linear Hypotheses:
##            Estimate Std. Error t value Pr(>|t|)
## B - A == 0   1.9373     0.3692   5.247  < 1e-04 ***
## C - A == 0   3.9992     0.3692  10.831  < 1e-04 ***
## D - A == 0   5.4693     0.3692  14.812  < 1e-04 ***
## C - B == 0   2.0619     0.3692   5.584  < 1e-04 ***
## D - B == 0   3.5320     0.3692   9.566  < 1e-04 ***
## D - C == 0   1.4701     0.3692   3.981  0.00172 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)

# Get simultaneous confidence intervals:
confint(myTukey, level = 0.95)

##
##   Simultaneous Confidence Intervals
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = value ~ group, data = tmp)
##
## Quantile = 2.6919
## 95% family-wise confidence level
##
##
## Linear Hypotheses:
##            Estimate lwr    upr
```

```
## B - A == 0 1.9373   0.9434 2.9313
## C - A == 0 3.9992   3.0052 4.9931
## D - A == 0 5.4693   4.4754 6.4633
## C - B == 0 2.0619   1.0679 3.0558
## D - B == 0 3.5320   2.5381 4.5260
## D - C == 0 1.4701   0.4762 2.4641

# Plot confidence intervals of the difference of means
plot(confint(myTukey), xlab = "Difference of means")
```

For Dunnett's test, the only additional step needed is defining the *reference* level of the experimental factor. By default, factor variables use the first factor (in alphabetical order) as the reference one, but you can use the base R function `relevel()` to change that (or function `fct_relevel()` from package `forcats`, if you want to force a specific ordering of levels instead of simply setting a reference).

```
# Make group C the reference one.
tmp$group = relevel(tmp$group, ref = "C")

# Run ANOVA
myAOV <- aov(value ~ group, data = tmp)
summary(myAOV)

##             Df Sum Sq Mean Sq F value   Pr(>F)
## group        3 171.37   57.12    83.8 2.65e-16 ***
## Residuals   36  24.54    0.68
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

### Dunnett test
myDun <- glht(myAOV, linfct = mcp(group = "Dunnett"))
summary(myDun)

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = value ~ group, data = tmp)
##
## Linear Hypotheses:
##            Estimate Std. Error t value Pr(>|t|)
```

```
## A - C == 0  -3.9992     0.3692 -10.831   <0.001 ***
## B - C == 0  -2.0619     0.3692  -5.584   <0.001 ***
## D - C == 0   1.4701     0.3692   3.981   <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)

# Get simultaneous confidence interval:
confint(myDun, level = 0.95)

##
##   Simultaneous Confidence Intervals
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = value ~ group, data = tmp)
##
## Quantile = 2.4517
## 95% family-wise confidence level
##
##
## Linear Hypotheses:
##            Estimate lwr     upr
## A - C == 0 -3.9992  -4.9045 -3.0939
## B - C == 0 -2.0619  -2.9671 -1.1566
## D - C == 0  1.4701   0.5649  2.3754

plot(confint(myDun), xlab = "Difference of means")
```

Given the statistically significant results obtained in the ANOVA performed in item **d**, you will now perform a post-hoc test to identify which strategies are different from which others, by how much, and which one is the best. You will use the Tukey method for all-vs-all comparisons.

For that, you will first need to make sure that your experimental factor (variable `strategy`) is coded as a `factor`, then re-fit your anova model and finally perform the sequential test.

e. Update your dataframe `df.strat` so that variable `strategy` is a factor variable, then run the ANOVA again using this updated dataframe and overwrite your old `aov.strat` with the new ANOVA results. Then run the multiple comparisons using function `glht()`, passing this new `aov.strat` and option `linfct = mcp(strategy`

= "Tukey"). Store the result of your call to `glht()` as object `strat.tukey`. Print `strat.tukey` to the console and check it to see which comparisons are statistically significant at the (already corrected) $\alpha = 0.05$ level.

Since you had 6 strategies, the Tukey test resulted in $(6 \times 5)/2 = 15$ comparisons, which is hard to interpret as a table. It may be easier to visualise it as a plot. For that, I'll show you how to calculate and plot the 95% confidence intervals using ggplot, highlighting the comparisons that involve the strategy with the highest point estimate of mean revenue. Check the code below, understand what's being done, then run it on your own console and check the results. In particular, notice how comparisons that are not statistically significant correspond to confidence intervals that intercept the zero line. From the results showin in this plot, you'll also be able to conclude that strategy F seems to be the best one, with Strategy E being a close enough contender.

```
# Calculate 95% confidence intervals and extract only the CIs
strat.tukey.cis <- confint(strat.tukey)$confint %>%
  as.data.frame()

# Add variables with the names of the levels involved in each comparison
# (based on the inherited rownames)
strat.tukey.cis <- strat.tukey.cis %>%
  mutate(Cmp   = rownames(strat.tukey.cis))

# Check which strategy had the best point estimate of performance
isbest <- df.strat.stats$strategy[which.max(df.strat.stats$meanRev)]

# Add a variable highlighting comparisons involving the strategy with the best
# point estimate
strat.tukey.cis <- strat.tukey.cis %>%
  mutate(HasBest = grepl(isbest, Cmp))

# Generate plot
ggplot(strat.tukey.cis,
       aes(x = Estimate, xmin = lwr, xmax = upr,
           y = Cmp,
           colour = !HasBest)) +
  geom_pointrange(show.legend = FALSE) +
  geom_vline(xintercept = 0, linetype = 3) +
  theme_light() +
  ylab("") +
  ggtitle("Pairwise comparisons of business strategies",
          "(Tukey Simultaneous 95% CIs)")
```

Blocked ANOVA

Finally, we'll touch on the generalisation of the paired t-tests for the case with multiple means being compared. The strategy used for this is called *blocking*, but the principle is the same: isolate the variability between different experimental units from the differences between levels of the experimental factor of interest.

Just like in the paired t-test case, the need for blocking appears whenever you have observations of all levels being collected under different *experimental units* (e.g., patients, problem instances, batches of raw materials etc). You want to focus your analysis on the differences between level means, and isolate ("block out") the variance emerging from differences between experimental units. Also, just like in the paired t-test case, each factor level should be represented by a single observation within each block (it there are multiple repeated measurements, aggregation by the mean is a valid strategy, as we'll do below). A *Complete Block Design* (CBD) consists of:

- one replicate per block (i.e., one observation of each factor level);
- independent blocks;
- independent within-block randomization (if needed - e.g., to prevent ordering effects)

The statistical model for the CBD is the same one as we used for the paired t-test, just accounting for more than 2 factor levels:

$$y_{ij} = \mu_i + \beta_j + \epsilon_{ij}, \begin{cases} i \in \{1, \dots, a\} \\ j \in \{1,2, \dots, N\} \end{cases}$$

Which can be written in as an effects model, just like the one-way ANOVA:

$$y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}, \begin{cases} i \in \{1, \dots, a\} \\ j \in \{1,2, \dots, N\} \end{cases}$$

The hypotheses of interest in of a CBD ANOVA are the same as the one-way anova, i.e.,

$$\begin{cases} H_0: \tau_i = 0, \ \forall \ i \\ H_1: \exists \tau_i \neq 0 \end{cases}$$

The R function used to run this test is the same `aov()` one, with just a small change in the formula used to include the blocking factor. It looks like:

```
aov(response ~ exp_factor + block_factor, data = ...)
```

Everything else (assumptions, post-anova tests etc) remains the same.

Q5: Comparison of multiple algorithms on multiple problems

For this question we'll use the same dataset from *Q3*, `UPMSP_SA_full_results.rds`, but now you'll run the full experimental comparison.

a. Load the dataset `UPMSP_SA_full_results.rds`. Use `group_by()` to group by both `Algorithm` and `Instance`, and then `summarise()` to make sure the performance of each algorithm on each instance is represented by a single number, representing the mean of the repeated measurements of `Makespan`. (*You can probably reuse a lot of the code from your Question 3a, but retaining all algorithms instead of only two*). Transform both the `Algorithm` and `Instance` variables into factor variables, and set the reference level of `Algorithm` as "Full". Store this dataset as `df.cbd`.

b. Perform an analysis of variance to test the differences of mean performance between the different algorithms, with `Instance` as a blocking variable. Use `summary()` to check if there are significant differences in the `Algorithm` factor. Assume that you are interested in a confidence level of 99%. For now, you can ignore the p-value associated with the blocking variable and focus only on the experimental factor.

c. Perform an all-vs.one comparison against the reference level (i.e., using the Dunnett method). Store the result of your call of `glht()` as variable `cbd.mht`.

d. Adapt the plotting function provided earlier to plot the confidence intervals of multiple post-anova comparisons to create a plot showing the results of this test (at the 99% confidence level). There is no need to colour by comparisons that involve the reference method (since in the all-vs-one comparisons all involve the reference). If you want (optional), try colouring instead by comparisons that were statistically significant.

Recall that in the context of this problem, smaller = better. Is there any method that was shown to be better than the reference one? Which ones are not significantly worse?