Arjun Krishnan

Day 04 Pre-class Assignment

Please submit your assignment as a RMarkdown or a Jupyter notebook. Include your code and results/plots for the programming exercises (1 & 3).

1. Calculating p-values using a permutation test

> This is *identical* to the exercise we did in-class tiday (Nov 13). I'm asking you to submit the code, results, and your interpretation.

- Simulate data on `number_of_days_with_cold` for two groups of 100 individuals each, one having taken a cold medicine and the other having taken a placebo.
  - Generate data from a normal distribution for each group with slightly different means (19.5 for group1 and 20 for group2) and the same standard deviation of 1.
- Define a test statistic equal to the difference between the means of the two groups. Calculate this statistic for the simulated data.
- Then, generate a null distribution of this test statistic by shuffling the data points between the two groups and calculating the test statistic each time, the whole thing repeated 10,000 times.
- Finally, compare the observed test statistic to this null distribution and get a p-value.

I would like you to:

- Write code to implement the above along with detailed comments and write descriptive text about what is going on at each stage.
- Make a plot of the null distribution (histogram) and mark the observed test statisitic on this histogram with a vertical line. Give the plot informative x- and y-axis labels and a title that also contains the p-value you calculated.
- Finally, define what a p-value in terms of this specific 'cold medicine' exercise.

2. Effect of effect size, sample size, and variance on the p-value.

Like we discussed in class, apart from the null hypothesis (which need not always be random chance), the p-value of a statistical test depends on:

1. Effect size,
2. Sample size, and
3. Variance within each group

To explore how these factors influence the p-value, I have written the R code below to simulate data for two groups (just like above) multiple times, each time varying the `effect_size`, `sample_size`, and `std_deviation` and calculating the p-value using a T-test.

```
# Explore the relationship between p-value, effect_size, sample_size, and within-group
variance

library(tidyverse)

ss_es_sd_pvalue <- {}

for(effect_size in seq(0.1, 1, 0.05)) {
for(sample_size in c(5, 10, 20, 50, 100, 200, 500, 1000)) {
for(std_deviation in c(0.5, 1, 2)) {

group1 <- rnorm(sample_size, mean = 0, sd = std_deviation)
group2 <- rnorm(sample_size, mean = effect_size, sd = std_deviation)

ttest_out <- t.test(group1, group2)

ss_es_sd_pvalue <- rbind(ss_es_sd_pvalue,
c(effect_size, sample_size, std_deviation, ttest_out$p.value))
}
}
}

colnames(ss_es_sd_pvalue) <- c("effect_size",
"sample_size",
"std_deviation",
"pvalue")
ss_es_sd_pvalue <- as.tibble(ss_es_sd_pvalue)

ss_es_sd_pvalue <- ss_es_sd_pvalue %>%
mutate(pvalue_below_thresh = (pvalue < 0.05))

plot_ss_es_sd_pvalue <- ss_es_sd_pvalue %>%
ggplot(aes(x = sample_size,
y = effect_size,
size = 2,
color = pvalue_below_thresh,
shape = pvalue_below_thresh)) +
geom_point() +
scale_shape_manual(values = c(4, 16)) +
scale_color_manual(values = c("blue", "red")) +
scale_x_log10() +
facet_wrap(~std_deviation, nrow = 1) +
theme_bw() +
theme(legend.position = "none")

ggsave(filename = "samplesize_effectsize_variance_pvalue.pdf",
plot_ss_es_sd_pvalue, height = 5, width = 12)
```

You can find the plot produced by this code here: https://drive.google.com/open?id=1CkRHlcBBPuoaSplt18XHqljTKzqF_M8L

Your tasks are the following:

- Carefully examine and annotate the code by writing detailed comments at each step.

- Examine the figure in the linked PDF and write a short paragraph about you observations on how these three quantities influence the p-value.
    - The figure contains 3 plots, one for each `std_deviation` value `(0.5, 1, 2)`. In each plot, the `sample_size` values are along the x-axis and the `effect_size` values are along the y-axis. Each point is either a red dot or a blue cross depending on whether the p-value for the t-test between the two groups (for that combination of `effect_size`, `sample_size`, and `std_deviation`) is below 0.05 or not.

3. P-hacking

> P-hacking is the practice of collecting or selecting data or statistical analyses until nonsignificant results become significant.

Write code to implement the following p-hacking process.

- Step 1: Simulate 20 values from a standard normal distribution with `mean = 0` and `var = 1`.
- Step 2: Randomly split these 20 values into two groups.
- Step 3: Use a t-test to calculate a p-value.
- Repeat Step 2 and Step 3 until the p-value is < 0.05.
    - (Don't do this manually; Write code that loops through steps 2 & 3 until p-value < 0.05.)

Run this code 5 times and report the number of times you had to loop until you got a p-value < 0.05 each time.

- Describe your thoughts on what this coding exercise has to do with p-hacking.