“I confirm that the following design is my own work, except where clearly indicated.”

Step 1: Preparation for simulation study

select the dataset

⬇

data formatting process

(select, combine and save the data of the two groups – CO_2 and N_2O)

⬇

explore the properties of the data

(statistical summary, assess normality and distribution)

⬇

confirm the research question and hypotheses:

“On average, is the concentration of CO_2 greater than the concentration of N_2O?”

H_0: “there is no difference between the mean of concentration of CO_2 and that of N_2O”.

H_1: “the mean of concentration of CO_2 is greater than that of N_2O”.

⬇

Choose two statistical tests (one parametric test is t-test, one non-parametric test is Mann-Whitney U test)

Step 2: Confirm the simulation scenarios

Scenario one ➡ examine the effect of sample sizes on size and power of two tests

① simulated sample size of group 1 is small vs group 2 is large

(e.g. $n1 = 10$ vs $n2 = 50$ or 1000)

② simulated sample size of group 1 is equal to group 2 but they are increasing gradually

(e.g. $n1 = n2 = 10$ & $n1 = n2 = 1000$)

Scenario two ➡ examine the effect of measurement error presented in the simulated data on size and power of two tests

(the reported values are always rounded up to the nearest integer)

Scenario three ➡ examine the effect of standard derivations of the simulated data on size and power of two tests

① standard derivation of simulated group 1 is small vs group 2 is large

(e.g. $sd1 = 1$ vs $sd2 = 1$ or 50)

② standard derivation of simulated group 1 is equal to group 2 but they are increasing gradually

(e.g. $sd1 = sd2 = 10$ & $sd1 = sd2 = 50$)

Scenario four ⟹ examine the effect size of simulated data on power of two tests (under different sample size n where $n1 = n2 = n$, explore the power of the two tests when the true difference between means is from small to large)

further combine scenario one and four to explore the required sample size to achieve the power of 1 given fixed standard derivation when effect size is smaller than 1.

Step 3: Generate and analyse simulated data

<u>General strategy:</u>

Write code to generate simulated data under four scenarios

⬇

Apply the two chosen statistical tests to the simulated data

⬇

Calculate the size and power of these statistical tests under each of the scenario

<u>Main planned functions:</u>

Initially separate functions for t-test and Mann-Whitney U test under four scenarios were considered; however there is only one line of code needed to change inside the function. This violate the modularity and dry principles. In this case, I reconsidered the design and add if-statement to specify the chosen tests. As a result, one function to generate simulation, calculate size and power under the chosen tests across four scenarios was constructed.

Aiming to combine the three steps in general strategy.
Function name: **simulate_test()**
Inputs:

sc : scenarios (1 or 2 or 3 or 4) (default as 1)
n1 : simulated sample size of group 1
n2 : simulated sample size of group 2
mean1 : mean of group 1 for simulated sample
mean2 : mean of group 2 for simulated sample
sd1 : standard derivation of group 1 for simulated sample
sd2 : standard derivation of group 2 for simulated sample
nsim : number of simulation (default as 1000)
H1 : alternative hypothesis ("less", "two.sided", default as "greater")
test: statistical tests we wish to examine ("wilcox.test", default as "t.test")

Processes:

1. Input Check
2. create “empty” p-values to store later
3. start of the for-loop

for each number of simulation:

a. set if-statement for difference scenarios
b. for scenario 1/3/4, use rnorm() to simulate data with specified n, mean and sd
   for scenario 2, use round(rnorm()) to simulate data with input n, mean and sd
c. set above simulated data as dataframes and we need to combine two dataframes
d. set if-statement to apply the t-test and Mann-Whitney U test to simulated data, and get p-value directly

repeat the above process nsim times (default as 1000 times) and get 1000 p-values

4. the size of the test can only be calculated when effect size is zero, thus it is worthy to emphasis on effect size. Here, calculate effect size
5. calculate size and power of tests under difference situations using if-statement (we set alpha level as 5%)
   if effect size is 0, size could be calculated as sum(p-values <= 0.05) / number of simulation; if effect size is not 0, power could be calculated as sum(p-values <= 0.05) / number of simulation.

Outputs:

one dimensional array: effect size (mu1 - mu2) and size or power of related tests

Input test function of simulate_test().

Function name: **simulate_input_test()**

Inputs:

sc : scenarios (1 or 2 or 3 or 4)
n1 : simulated sample size of group 1
n2 : simulated sample size of group 2
mean1 : mean of group 1 for simulated sample
mean2 : mean of group 2 for simulated sample
sd1 : standard derivation of group 1 for simulated sample
sd2 : standard derivation of group 2 for simulated sample
nsim : number of simulation
H1 : alternative hypothesis
test : statistical tests we wish to examine

Processes:

if statements

1. check scenario number (sc) should be 1 or 2 or 3 or 4
2. check name of the two chosen tests should be “t.test” and “wilcox.test”
3. check number of simulations (nsim) should be numeric
4. check specified H1 should be exactly “less” or “greater” or “two.sided”
5. check sample size (n1, n2), number of simulation (nsim) should be positive; and standard derivations (sd1, sd2) should bigger than zero
6. check scenario number (sc), sample sizes (n1, n2) and number of simulation (nsim) should be integer

Outputs:

returns FALSE if the required properties are not met, TRUE otherwise

Output test function of simulate_test().
Function name: **simulate_output_test()**
Inputs:

sc : scenarios (1 or 2 or 3 or 4) (default as 1)
n1 : simulated sample size of group 1
n2 : simulated sample size of group 2
mean1 : mean of group 1 for simulated sample
mean2 : mean of group 2 for simulated sample
sd1 : standard derivation of group 1 for simulated sample
sd2 : standard derivation of group 2 for simulated sample
nsim : number of simulation (default as 1000)
H1 : alternative hypothesis ("less", "two.sided", default as "greater")
test: statistical tests we wish to examine ("wilcox.test", default as "t.test")

Processes:

1. run simulate_test function and name it as out
2. calculate effect size
3. check if output length is 2
4. check if effect size is equal to the first output
5. check if calculated size or power is between 0 and 1

Outputs:

print "PASSED" if outputs matches with expected outputs and give details
print "FAILED" if outputs don't match the expected one and give details

Step 4: Analyse size and power of chosen tests under each scenario

Analyse size and power by plugging in the numbers to **simulate_test** function under each scenario

⬇

Write code to visualize the power and size of each chosen test under each scenario

⬇

Analyse and summarize the findings

Difference in code between t-test and Mann- Whitney U test ➡ if Mann-Whitney U test is used, we need to add test = "wilcox.test" as input in simulate_test function; we don't need to add any input to specify the t-test since the default is test = "t.test". Otherwise, all the rest of the inputs are the same for both of the tests.

For scenario one, two and three, to calculate the size of the tests, we had mean1 = mean2 = 282.1. To calculate the power of the tests, we had mean1 = 282.1, mean2 = 268.5.

Except for scenario one case 1 to examine the impact of unequal sample sizes, for the rest of scenarios we considered equal sample sizes based on the properties of the original data. All analysis within each scenario was considered under the same conditions of sample sizes, means and standard derivations. Details as follows:

Scenario one ⟹ explore the effect of sample size

① $n1$ small vs $n2$ large

size: sc = 1, n1 = 10, n2 = 50 or 1000, mean1 = 282.1, mean2 = 282.1, sd1 = 11, sd2 = 7

power: sc = 1, n1 = 10, n2 = 50 or 1000, mean1 = 282.1, mean2 = 268.5, sd1 = 11, sd2 = 7

plots (difference between n1 and n2 is increasing): n1 = 10, n2 = from 10 to 1000, sd1 = 11, sd2 = 7, increment = width = 10

② $n1 = n2$ but increasing

size: sc = 1, n1 = 10 or 1000, n2 = 10 or 1000, mean1 = 282.1, mean2 = 282.1, sd1 = 11, sd2 = 7

power: sc =1 n1 = 10 or 1000, n2 = 10 or 1000, mean1 = 282.1, mean2 = 268.5, sd1 = 11, sd2 = 7

plots (n1 and n2 are increasing) : from n1 = n2 = 10 to 1000, sd1 = 11, sd2 = 7, increment = width = 10

Scenario two ⟹ explore the effect of measurement error presented in simulated data

with measurement error:

size: sc = 2, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 282.1, sd1 = 11, sd2 = 7

power: sc = 2, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 268.5, sd1 = 11, sd2 = 7

without measurement error:

size: sc = 1, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 282.1, sd1 = 11, sd2 = 7

power: sc = 1, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 268.5, sd1 = 11, sd2 = 7

plots for calculating size and power 100 times under scenario 2 : round = 100, n1 = n2 = 100, sd1 = 11, sd2 = 7, increment = width = 1

Scenario three ⟹ explore the effect of standard derivation

① sd1 small vs sd2 large

size: sc = 3, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 282.1, sd1 = 1, sd2 = 10 or 50

power: sc = 3, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 268.5, sd1 = 11, sd2 = 10 or 50

plots (difference between sd1 and sd2 is increasing) : sd1 = 11, sd2 = from 12 to 100, n1 = n2 = 100, increment = width = 1

② sd1 = sd2 but increasing

size: sc = 3, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 282.1, sd1 = 10 or 50, sd2 = 10 or 50

power: sc =3, n1 = 100, n2 = 100, mean1 = 282.1, mean2 = 268.5, sd1 = 10 or 50, sd2 = 10 or 50

plots (sd1 and sd2 are increasing) : from sd1 = sd2 = 10 to 100, n1 = n2 = 100, increment = width = 1

Scenario four ⟹ explore the impact of effect size on power of the tests general analysis:

effect size is 0.5:

sc = 4, n1 = 100, n2 = 100, mean1 = 269, mean2 = 268.5, sd1 = 11, sd2 = 7

effect size is 1.5:

sc = 4, n1 = 100, n2 = 100, mean1 = 270, mean2 = 268.5, sd1 = 11, sd2 = 7

effect size is 11.5:

sc = 4, n1 = 100, n2 = 100, mean1 = 280, mean2 = 268.5, sd1 = 11, sd2 = 7

plots of effect size increasing from 1 to 20 given increasing sample size (n1 = n2 = 10, 100 and 1000) (mean1 = 282.1 + i, mean2 = 282.1, sd1 = 11, sd2 = 7, increment = width = 1)

analysis of small effect size given sd1 = 11, sd2 = 7:

effect size is 0.2:

sc = 4, n1 = 10000, n2 = 10000, mean1 = 268.7, mean2 = 268.5, sd1 = 11, sd2 = 7

plots of power as n1 = n2 = n increases from 50 to 1000 when effect size is 0.2 (mean1 = 268.7, mean2 = 268.5) / 0.5 (mean1 = 269, mean2 = 268.5) / 0.8 (mean1 = 269.3, mean2 = 268.5) given sd1 = sd2 = 1 (increment = width = 50)

Planned functions for graphs:

Initially one function to combine all graphs for the four scenarios was considered; however, considering the following reasons and drawbacks, I redesigned the process and combined the necessary steps according to their functionality across the four scenarios.

1. *one function for graphs involves lots of if-statements and we need to show lots of inputs under difference scenarios.*
2. *it took really long time to run the function when we give large number for the inputs. For example, we have sample size of 1000 as inputs.*
3. *it is not user-friendly. Users can easily get confused about which inputs they should put and what they could get.*
4. *it violates modularity principles since it is not easy to reuse and not easy for maintenance.*

Thus, considering the efficiency, user-friendly and ways to speed up code, I grouped some features together and designed four functions for creating the plots according to their functionality. In this case, modularity and dry principle are not violated basically. Users could easily reproduce the graph based on chosen scenarios. Even though the steps for getting the plots are similar, there are essential differences within each step between them. In addition, we could reduce inputs and could easily fix the errors based on scenarios. I further added width (increment of the sequence) as an input in order to be flexible and speed up code.

Note: if chosen width is small, the function may need some time to run.

Aiming to examine power and size when
sc1 : difference between n1 and n2 is increasing
sc3 : difference between standard derivations is increasing
Function name: **d_increase()**
Inputs:

- sc : scenario (1 or 3)
- d_lower : lower bound of difference (set default as 1)
- d_upper : upper bound of difference (set default as 10)
- width : number (increment of the sequence) (set default as 10)
- mean1 : mean of group 1 for simulated sample
- mean2 : mean of group 2 for simulated sample
- test : statistical tests we wish to examine ("wilcox.test", default as "t.test")

Processes:

1. Input Check
2. create an empty vector to store calculated power or size
3. create sequence (with increment of the sequence)
4. start the for-loop
   use if-statement:
   if scenario is 1, use simulate_test function to calculate size or power for specified test, particularly $n2 = n1 + i$ $(n1 = 10, sd1 = 11, sd2 = 7)$
   if scenario is 3, use simulate_test function to calculate size or power for specified test, particularly $sd2 = sd1 + i$ $(sd1 = 11, n1 = n2 = 100)$
5. consider the case that d_lower is not 1, we delete the empty numbers inside the vector, then create a dataframe of differences and size or power
6. calculate effect size
7. use multiple if-statements to get plots under different situations

Outputs:

return plot.
If sc = 1 : x = difference between n1 and n2, y = power / size of t_test / mann_test
If sc = 3 : x = difference between sd1 and sd2, y = power / size of t_test / mann_test

Aiming to examine power and size when
sc1 : n increasing
sc3 : standard derivation increasing
sc4 : evaluate the impact of small effect size on sample sizes
Function name: **increase()**
Inputs:

- sc : scenario (1 or 3 or 4)
- lower : lower bound (set default as 10)
- upper : upper bound (set default as 11)
- width : number (increment of the sequence) (set default as 10)
- mean1 : mean of group 1 for simulated sample
- mean2 : mean of group 2 for simulated sample
- test : statistical tests we wish to examine ("wilcox.test", default as "t.test")

Processes:

1. Input Check
2. create an empty vector to store calculated power or size
3. create sequence (with increment of the sequence)
4. start the for-loop
   use if-statement:
   if scenario is 1, use simulate_test function to calculate size or power for specified test, particularly $n1 = n2 = i$ (sd1 = 11, sd2 = 7)
   if scenario is 4, use simulate_test function to calculate power for specified test, particularly $n1 = n2 = i$ (sd1 = sd2 = 1)
   if scenario is 3, use simulate_test function to calculate size or power for specified test, particularly $sd1 = sd2 = i$ (n1 = n2 = 100)
5. consider the case that lower is not 1, we delete the empty numbers inside the vector, then create a dataframe of i and size or power
6. calculate effect size
7. use multiple if-statements to get plots under different situations

Outputs:

return plot.
If sc = 1 : x = sample size n, y = power / size of t_test / mann_test
If sc = 4 : x = sample size n, y = power of t_test / mann_test
If sc = 3 : x = standard derivation, y = power / size of t_test / mann_test

Aiming to examine power and size when there is measurement error presented in simulated data under sc2

Function name: **measure_error()**

Inputs:

n1 : sample size for group 1 (set default as 100)
n2 : sample size for group 2 (set default as 100)
mean1 : mean of group 1 for simulated sample
mean2 : mean of group 2 for simulated sample
test : statistical tests we wish to examine ("wilcox.test", default as "t.test")
round : number of times we run the scenario 2 (set default as 20)
width : number (increment of the sequence) (set default as 1)

Processes:

1. Input Check
2. create an empty vector to store calculated power
3. create sequence (with increment of the sequence)
4. start the for-loop
   use simulate_test function to calculate power or size for specified test, particularly using sc = 2 (sd1 = 11, sd2 = 7)
5. create a dataframe of round and size or power
6. calculate effect size
7. use if-statements to get plots under chosen two tests

Outputs:

return plot: x = round, y = power / size of t_test / mann_test

Aiming to examine power when
sc4 : effect size (mean1 – mean2) increasing
Function name: **effect_size_increase()**
Inputs:

lower : lower bound (set default as 1)
upper : upper bound (set default as 10)
width : number (increment of the sequence) (set default as 1)
n : sample size (n1 = n2 = n)
H1 : alternative hypothesis ("less", "two.sided", default as "greater")
test : statistical tests we wish to examine ("wilcox.test", default as "t.test")

Processes:

1. Input Check
2. create an empty vector to store calculated power
3. create sequence (with increment of the sequence)
4. start the for-loop
   use simulate_test function to calculate power for specified test, particularly $mean1 = 282.1 + i,\ mean2 = 282.1$ (n1 = n2 = n, sd1 = 11, sd2 = 7)
5. consider the case that lower is not 1, we delete the empty numbers inside the vector, then create a dataframe of i and size or power
6. use if-statements to get plots under chosen two tests

Outputs:

return plot: x = effect size, y = power of t_test / mann_test