Statistical Analysis

Gabriele Russo

September 2024

1 Introduction

This report presents a statistical analysis comparing my implementation of the Research Track 1 first assignment (AssignmentRussoGabriele.py) with Professor Recchiuto's implementation (assignment1.py). The assignment scenario is described in detail in this GitHub repository: Click here to visit the github repository.

The goal of this analysis is to test the null hypothesis, which states that the two implementations exhibit the same performance.

To gather data for the statistical analysis, I developed a Python script (`run_multiple_times.py`), that sequentially runs both implementations 30 times each (a total of 60 executions). For every execution, both implementations operate under the same conditions, specifically referring to the number and placement of silver tokens in the circuit. However, the placement and number of silver tokens are varied across the 30 simulations to ensure that each scenario is unique.

2 Lilliefors test

Before starting with the statistical analysis, I have to check if the data gathered belog to the normal distribution, and since I do not know the mean and the standard deviation of the population I need to use the Lilliefors test.

2.1 How to perform the Lilliefors test

In order to apply the lillifors test to the data, we have to follow the following steps:

1. **Compute the mean and standar deviation of the sample data**: this is needed because the Lilliefors test use these estimated values from the sample to construct the theoretical normal distribution to which the data can be compared

2. **Construct the empirical cumulative distribution function (CDF) of the data**: the empirical CDF is the function that, for each value in

the sample, indicates the proportion of data that are less than or equal to that value.

3. **Compare the empirical CDF with the CDF of the theoretical normal distribution**: the comparison is made by finding the maximum absolute difference between the empirical CDF and the theoretical one.

4. **Compute the value of the Lilliefors test, known as the D-statistic**: this value represents the maximum difference between the empirical and theoretical CDF.

5. **Compare the D-statistic with the Lilliefors critical value**: we can have the following two results.

   - D-statistic is greater than the critical value (or the p-value is less than the significance level, typically 0.05: this result means that the sample data do not follow the normal distribution
   - D-statistic is less than or equal to the critical value (or the p-value is above the significance level): this result means that there is not enough evidence to say that the data does not follow a normal distribution.

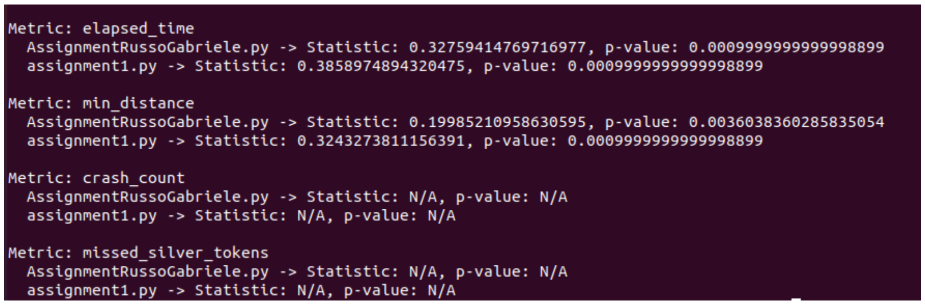
2.2 Lilliefors test Results

Since I need to compare the two implementations of the first assignment in the Research Track 1 course, it is better to divide the data into separate samples for each implementation and then perform the Lilliefors test on each. This approach should make the statistical analysis more robust. In order to perform the Lilliefors test I have implemented a python script called `Lilliefors_Test.py` (you can find it in the **tests** folder). The script gives as output the results of Lilliefors test for each metric of the two implementations, and they are shown in the figure 1.

Since the **p-value** is consistently lower than the **significance level** (0.05) for each metric (except for the crash count and missed silver tokens, which are always zero), we can reject the null hypothesis of the Lilliefors test. This indicates that the sample data does not follow a normal distribution. As a result, we must use a **non parametric test** for the statistical analysis.

3 Wilcoxon-Mann-Whitney test (or U test)

The Mann-Whitney U test is a non-parametric statistical test used to compare two independent samples to determine whether one tends to have significantly larger or smaller values than the other. In our case, this is the right choice because we have discovered that the sample data are not normally distributed. Furthermore, each observation in one group is independent of those in the other group, as each simulation of one implementation is independent of the other. Also in this test we have the null hypothesis and the alternative hypothesis:



```
Metric: elapsed_time
  AssignmentRussoGabriele.py -> Statistic: 0.32759414769716977, p-value: 0.0009999999999998899
  assignment1.py -> Statistic: 0.3858974894320475, p-value: 0.0009999999999998899

Metric: min_distance
  AssignmentRussoGabriele.py -> Statistic: 0.19985210958630595, p-value: 0.0036038360285835054
  assignment1.py -> Statistic: 0.3243273811156391, p-value: 0.0009999999999998899

Metric: crash_count
  AssignmentRussoGabriele.py -> Statistic: N/A, p-value: N/A
  assignment1.py -> Statistic: N/A, p-value: N/A

Metric: missed_silver_tokens
  AssignmentRussoGabriele.py -> Statistic: N/A, p-value: N/A
  assignment1.py -> Statistic: N/A, p-value: N/A
```

Figure 1: Lilliefors test results

- **Null Hypothesis** (H_0): The results from both implementations have the same distribution, meaning there is no significant difference in performance between the two implementations.
- **Alternative Hypothesis** (H_a): The distributions of the results from the two implementations are different, indicating a significant difference in performance between the two implementations.

3.1 How to perform the U test

This test compares the ranks of the data points, where Ranks are simply the positions of the data points when they are ordered from the smallest to the largest value. By using ranks, the test focuses on the relative positions of the data points rather than their actual values. This allows the test to detect whether one group generally has higher or lower values than the other. We have to follow the following steps in order to perform the U Test:

1. Combine the two samples into a single group of data.
2. Sort all the data in ascending order and assign a rank (position) to each data point.
3. Calculate the sum of ranks for each group.
4. Calculate the **U statistic**, which is based on the sum of the ranks. The U statistic represents the number of times a value from the first group is greater than a value from the second group.
5. Compare the U value with a reference distribution to determine the **p-value**, which tells you whether to reject or accept the null hypothesis.

The **U statistic** is computed as follow:

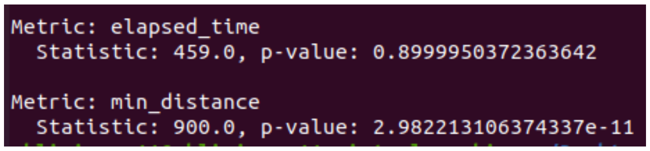
$$U_1 = n_1 \times n_2 + \frac{n_1(n_1+1)}{2} - R_1,$$
$$U_2 = n_1 \times n_2 + \frac{n_2(n_2+1)}{2} - R_2,$$

where n_1 and n_2 are the sample sizes of the two groups, and R_1 and R_2 are the sums of ranks for each sample. The final U statistic is **the minimum of U_1 and U_2**.

3.2 U-Test Results

In order to perform the Mann-Whitney U test I have implemented a python script called `U_Test.py` (you can find it in the **tests** folder). The script gives as output the results of U-Test for each metric ($elapsed_time$ and $min_distance$), and they are shown in the figure 2.

Therefore, the Mann-Whitney U test indicates no significant difference in `elapsed_time` between the two implementations (U = 459.0, p = 0.8999) because the p-value is higher than the significance level (0.05). However, a significant difference was found in `min_distance` (U = 900.0, $p < 0.001$), suggesting that one implementation consistently results in a different minimum distance compared to the other. We have to investigate further on the `min_distance`



```
Metric: elapsed_time
  Statistic: 459.0, p-value: 0.8999950372363642

Metric: min_distance
  Statistic: 900.0, p-value: 2.982213106374337e-11
```

Figure 2: U-Test results

Since the null hypothesis is rejected for the minimum distance, this indicates a difference between the two implementations. However, further investigation is needed to determine which implementation is better. To explore this, we will use a graphical method, specifically, a histogram, as shown in Figure 3.

3.2.1 Interpretation of the Histogram

We consider firstly the distribution of the minimum distance for the AssignmentRussoGabriele.py (Blue Bars):

- The distribution is more spread out, ranging from about 0.60 to 0.75.
- The peak frequency appears around the higher end of the distribution, suggesting that this implementation generally keeps a larger distance from obstacles.

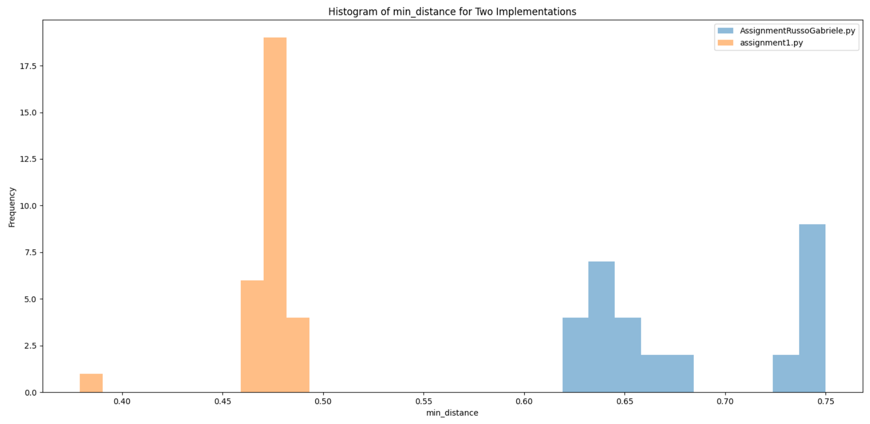


Figure 3: min distance histogram

Secondly, we consider the distribution of the minimum distance for the assignment1.py (Orange Bars):

- The distribution is much more concentrated, between 0.40 and 0.50
- There is a clear peak around 0.45, indicating that this implementation consistently navigates closer to obstacles.

Therefore, in the case of the AssignmentRussoGabriele.py, we have that the wider range and higher values for `min_distance` suggest that this implementation tends to maintain a safer, larger distance from obstacles, but with more variability in how close it gets to obstacles. Instead, in the case of the assignment1.py, we have that the narrow and lower range of `min_distance` values implies that this implementation is more consistent in its approach, maintaining a closer, more efficient proximity to obstacles.

4 Chi-square Test

The last metric that we have to compare is the `completed_circuit`, but in this case we have that the `completed_circuit` can be true or false, therefore it is not a numerical data but a **categorical data** (binary). In the case of categorical data we can perform the **Chi-Square Test**, which is a statistical method used to determine if there is a significant association between two categorical variables, in our case the implementation type (AssignmentRussoGabriele.py and assignment1.py) and the circuit completion status. The test compares the observed frequencies of these categories with the expected frequencies under the assumption that the variables are independent (i.e., there is no association between them). For our case, the goal is to determine whether the observed differences in their success rates are statistically significant or if they could be due to chance.

4.1 How to perform the Chi-Square test

In the following are the steps that we have to follow in order to perform the Chi-Square test:

1. Create the Contingency Table: The contingency table summarizes the observed frequencies of success and failure for each algorithm
2. Calculate the Expected Frequencies: Under the null hypothesis (H_0), the success of an algorithm is independent of which algorithm is used. The expected frequencies assume that the overall success and failure rates are equally distributed across both algorithms, therefore we have:
3. Compute the **Chi-Square statistic**: The Chi-Square statistic is calculated using the formula

$$\chi^2 = \sum \frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}}$$

4. Compare the Chi-Square Statistic to the Critical Value: The calculated χ^2 value is compared to a critical value from the Chi-Square distribution table, which depends on the degrees of freedom and the chosen significance level (commonly 0.05).

Therefore, we start with the creation of the Contingency table, using the python script `contingency_table.py` that you can find in the **tests** folder. Once we have the contingency table, shown in the figure 4, we can compute the expected frequencies as follow:

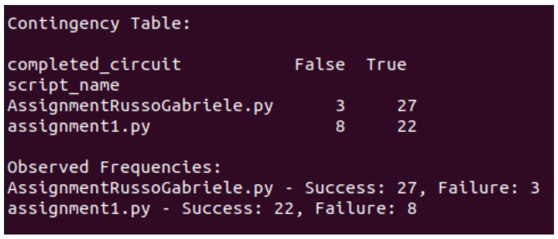
- For the AssignmentRussoGabriele.py (algorithm 1):

$$\text{Expected Success} = \frac{(\text{Total Success} \times \text{Total for Algorithm 1})}{\text{Grand Total}} = \frac{49 \times 30}{60} = 24.5$$

$$\text{Expected Failure} = \frac{(\text{Total Failure} \times \text{Total for Algorithm 1})}{\text{Grand Total}} = \frac{11 \times 30}{60} = 5.5$$

- For the assignment1.py (algorithm 2): since, we consider the null hypothesis, Algorithm 1 has the same performance of Algorithm 2, therefore the results of the Expected frequencies of the Algorithm 1, are the same of the algorithm 2.

Now we can compute the Chi-Square statistic:



```
Contingency Table:

completed_circuit           False  True
script_name
AssignmentRussoGabriele.py      3     27
assignment1.py                  8     22

Observed Frequencies:
AssignmentRussoGabriele.py - Success: 27, Failure: 3
assignment1.py - Success: 22, Failure: 8
```

Figure 4: Contingency Table

- For the AssignmentRussoGabriele.py (algorithm 1):

  Success:
  $$\frac{(27-24.5)^2}{24.5} = 0.255$$
  Failure:
  $$\frac{(3-5.5)^2}{5.5} = 1.136$$

- For the assignment1.py algorithm 2):

  Success:
  $$\frac{(22-24.5)^2}{24.5} = 0.255$$
  Failure:
  $$\frac{(8-5.5)^2}{5.5} = 1.136$$

Therefore, the Chi Square statistic is:

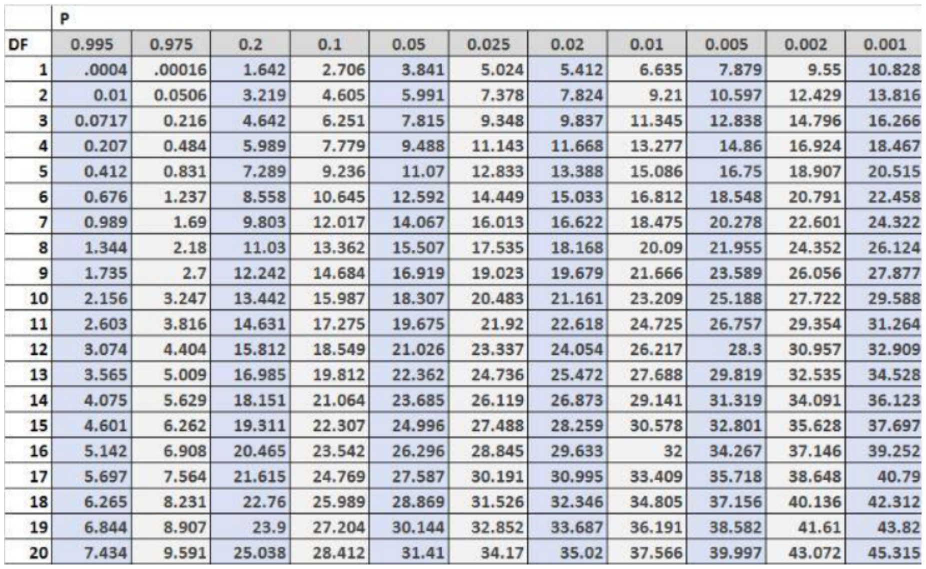
$$\chi^2 = \sum \frac{(\text{Observed} - \text{Expected})^2}{\text{Expected}} = 2.782$$

The Degrees of freedom for the test is calculated as:

$$\text{df} = (\text{number of rows} - 1) \times (\text{number of columns} - 1)$$

In our case it is 1, so we have that $\chi^2 = 2.782$ and $df = 1$. Therefore, now we can find the associated p-value using the table in the figure 5

The associated p-value is between 0.1 and 0.05, therefore we can not reject the null Hypothesis. This means that we do not have sufficient evidence to conclude that there is a significant difference between the success rates of AssignmentRussoGabriele.py and assignment1.py, suggesting that any observed difference could be due to random variation rather than a true difference in performance between the two implementations.



	P										
DF	0.995	0.975	0.2	0.1	0.05	0.025	0.02	0.01	0.005	0.002	0.001
1	.0004	.00016	1.642	2.706	3.841	5.024	5.412	6.635	7.879	9.55	10.828
2	0.01	0.0506	3.219	4.605	5.991	7.378	7.824	9.21	10.597	12.429	13.816
3	0.0717	0.216	4.642	6.251	7.815	9.348	9.837	11.345	12.838	14.796	16.266
4	0.207	0.484	5.989	7.779	9.488	11.143	11.668	13.277	14.86	16.924	18.467
5	0.412	0.831	7.289	9.236	11.07	12.833	13.388	15.086	16.75	18.907	20.515
6	0.676	1.237	8.558	10.645	12.592	14.449	15.033	16.812	18.548	20.791	22.458
7	0.989	1.69	9.803	12.017	14.067	16.013	16.622	18.475	20.278	22.601	24.322
8	1.344	2.18	11.03	13.362	15.507	17.535	18.168	20.09	21.955	24.352	26.124
9	1.735	2.7	12.242	14.684	16.919	19.023	19.679	21.666	23.589	26.056	27.877
10	2.156	3.247	13.442	15.987	18.307	20.483	21.161	23.209	25.188	27.722	29.588
11	2.603	3.816	14.631	17.275	19.675	21.92	22.618	24.725	26.757	29.354	31.264
12	3.074	4.404	15.812	18.549	21.026	23.337	24.054	26.217	28.3	30.957	32.909
13	3.565	5.009	16.985	19.812	22.362	24.736	25.472	27.688	29.819	32.535	34.528
14	4.075	5.629	18.151	21.064	23.685	26.119	26.873	29.141	31.319	34.091	36.123
15	4.601	6.262	19.311	22.307	24.996	27.488	28.259	30.578	32.801	35.628	37.697
16	5.142	6.908	20.465	23.542	26.296	28.845	29.633	32	34.267	37.146	39.252
17	5.697	7.564	21.615	24.769	27.587	30.191	30.995	33.409	35.718	38.648	40.79
18	6.265	8.231	22.76	25.989	28.869	31.526	32.346	34.805	37.156	40.136	42.312
19	6.844	8.907	23.9	27.204	30.144	32.852	33.687	36.191	38.582	41.61	43.82
20	7.434	9.591	25.038	28.412	31.41	34.17	35.02	37.566	39.997	43.072	45.315

Figure 5: Table to find the p-value for the Chi-Square Test

5 Conclusions

After all the test that we have done in the previous sections, we can conclude that in general there are no significant differences between the two implementations, the only difference is in the minimum distance from the obstacle. Therefore, We can claim that the AssignmentRussoGabriele.py implementation tends to maintain a safer, larger distance from obstacles, but with more variability in how close it gets to obstacles. Instead, the assignment1.py implementation, is more consistent in its approach, maintaining a closer proximity to obstacles. Therefore, we can not claim that this two implementations have the same performance overall, and if we want to say what implementation is better, it depends on the scenario where the implementation has to be used. Given that both implementations have zero collision rates, the narrower range of `min_distance` for assignment1.py suggests that this implementation might be optimized for navigating closer to obstacles without compromising safety, that can be useful in scenarios where the circuit is tight. Meanwhile, AssignmentRussoGabriele.py might prioritize maintaining a larger distance from obstacles, which could be beneficial in environments where safety is a concern.