CST4070 – Challenge A1 (Block 1)

Computing Pair-wise Pearson Correlations with P-Values

Objective: To practice generating random data, computing pair-wise Pearson correlations between variables, and determining the significance of correlations using -values.

Instructions:

1. Data Generation:

Generate 6 vectors, each containing 10 random elements. These random elements should follow a normal distribution.

2. Correlation Computation:

Calculate the Pearson correlation coefficients between all pairs of vectors. There will be a total of 6*6 correlations.

3. Printing Results:

Print the all the correlation coefficients and all P-values in a structured format.

4. Interpretation:

Interpret the results and discuss any correlations that are statistically significant (p - value less than 0.10). Discuss the importance of p-values in determining the significance of correlations. Do you get any statistically significant correlations? If so, how can you justify it given that the variables are random and should not have any correlation?

Sub-Challenge 1]

Generate 6 vectors, each containing 10 random elements. These random elements should follow a normal distribution.

a) **Method Overview:**

- We will generate 6 vectors from v1 to v6, and assign 10 random elements following a normal distribution pattern to each variable, we will add these vectors to a data frame for the ease of visualizing.

b) **Code/Output:**

- The code used for this section is as follows:

```
- N = 10
- set.seed(192)
- df=data.frame(v1 = rnorm(N),
-                v2 = rnorm(N),
-                v3 = rnorm(N),
-                v4 = rnorm(N),
-                v5 = rnorm(N),
-                v6 = rnorm(N))
```

- This results in the output:

	v1	v2	v3	v4	v5	v6
1	1.00505360	0.2172209	-0.4639815	0.04004817	-0.32298084	1.479478536
2	0.13264543	-0.8333133	-2.2226745	0.73408213	-0.85093408	-0.670203871
3	0.47957896	-0.2802616	0.4815448	-2.13976672	0.31478554	-0.009429853
4	-0.42460545	0.0899365	-1.1446767	-0.30933397	1.06830286	0.939244599
5	-1.06534651	1.4665229	0.4274639	1.17473766	1.71836351	-0.819524301
6	-0.36481801	2.1348569	-0.9328331	2.37555696	1.41132178	0.680253148
7	-0.39892715	-0.4953298	1.5309681	0.30956765	0.10225238	0.393596542
8	0.09942895	-0.1009662	-0.3485467	0.79310255	0.02510421	0.667619618
9	-0.23926583	1.9123056	0.1587976	0.33163986	-1.36963105	-0.756999355
10	0.19461431	-0.5218629	0.5640927	-0.28063212	-0.65697921	0.479084867

c and d) Explanation and Critical Analysis of results

Normal distribution follows a bell curve where a majority of the datapoints lie near the center with about 68% of results lying between one standard deviation (away or towards) the mean. From a random number generation standpoint, this theoretically equates to about 68% of our values being generated between -1 to +1 , 95% of our values being generated in between -2 to +2 and 99.7% of them being generated in between -3 to +3.

From our generated dataset we can see that 44 entries lie between -1 and +1, about 73.3%, which is close to the theoretical probability density of 68% and with a larger

dataset this value will tend closer to that standard. From this we can understand that our random values are *not pure random* but have a probability weightage attached to them following the standard probability density of a normal distribution curve.

As the probability curve is also symmetrical, we should expect to see about 50% of positive values and 50% of negative values in our dataset. What we observe is 34 out of 60 positive entries (around 56.7%, slightly higher but still close to the expected 50%)

e) In our code we created a variable N for number of data points in each vector, while all the variables have a N of 10 in this assignment, we create this variable for the sake of keeping the code generic and minimizing changes to the formulation if datapoints are added to the vectors.

Set.seed(192) is used to consistently generate the same data frame that was initially randomly generated so if either the data frame is lost, altered or the analysis is done in multiple sessions, the same dataset can be generated to analyse. (This line of code is not necessary if the data frame is stored or saved externally and can be reopened from a file) Any number can be used in the function as long as it is consistent to replicate the seed, I went with 192.

Rnorm is the function used to randomly generate N no. of values following the normal distribution probability density (takes one argument, the number of values to generate, which in our case is assigned to variable N – set as 10)

Sub-Challenge 2]

Correlation Computation: Calculate the Pearson correlation coefficients between all pairs of vectors. There will be a total of 6*6 correlations.

a) **Pearson coefficient is calculated using the following formula:**

Correlation Coefficient Formula

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

Where n is the number of data points, and X and Y are the two vector datasets we are comparing.

Initially I completed this calculation, along with t-value and p-value calculations between v1 and v2 utilizing excel to ensure the answers and methodology line up. Below is the excel calculations.

	1	2	3	4	5	6	7	8	9	10	SUMMATION	SUMMATION SQUARED
V1 [X]	1.005	0.133	0.480	-0.425	-1.065	-0.365	-0.399	0.099	-0.239	0.195	**-0.582**	**0.338**
V2 [Y]	0.217	-0.833	-0.280	0.090	1.467	2.135	-0.495	-0.101	1.912	-0.522	**3.589**	**12.882**
X^^2	1.010	0.018	0.230	0.180	1.135	0.133	0.159	0.010	0.057	0.038	**2.970**	**-**
Y^^2	0.047	0.694	0.079	0.008	2.151	4.558	0.245	0.010	3.657	0.272	**11.721**	**-**
XY	0.218	-0.111	-0.134	-0.038	-1.562	-0.779	0.198	-0.010	-0.458	-0.102	**-2.778**	
N (no of data pts)	10											
R VALUE	**-0.4641**											

Calculating the coefficient R value between v1 and v2 I've gotten -0.4641.

Once I've validated my method I worked on the code:

b) Code:

```
r = cor(df, method = c("Pearson"))
```

Output:

	v1	v2	v3	v4	v5	v6
v1	1.0000000	-0.46410291	-0.15479404	-0.48013741	-0.54872946	0.48367322
v2	-0.4641029	1.00000000	0.01308828	0.55083155	0.34166226	-0.21114102
v3	-0.1547940	0.01308828	1.00000000	-0.30089929	0.02369315	-0.03702225
v4	-0.4801374	0.55083155	-0.30089929	1.00000000	0.29106884	-0.06172388
v5	-0.5487295	0.34166226	0.02369315	0.29106884	1.00000000	0.15268933
v6	0.4836732	-0.21114102	-0.03702225	-0.06172388	0.15268933	1.00000000

c and d) Explanation and Critical Analysis of results

- As seen on the table, the Correlation coefficient between v1 and v2 matches the value received in our hands on/Excel calculation.
- There are a diagonal of 1's as the Pearson coefficient checks how each vector is correlated with another and a vector will always be fully correlated with itself by default
- This results in a 6x6 table with 15 valid correlation results (since 6 of them are correlating vectors with themselves and the other 15 are diagonally mirroring the results)
- The correlation coefficient ranges from -1 to +1 to signify perfect negative correlation and perfect positive correlation respectively and is mainly applicable on linear data.
- While most Correlation Coefficients are weak, some (such as v4 and v2) are relatively decent.
- When studying this data more, it seems that by chance a value >2 occurred on the 6th entry of both these columns which is comparatively quite a rare chance occurrence and is the main cause of the correlation coefficient being boosted.

e) Arguments taken by the function 'cor' include the data frame and the type of correlation (Pearson/spearmen) and returns a 6x6 data frame of Pearson coefficients

Sub-Challenge 3]

Printing Results: Print all the correlation coefficients and all P-values in a structured format.

a) Attaining the p-value from the Pearson coefficient requires the t-statistic to be calculated first, this can be done through the formula:

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}}$$

Where n is the number of data points per vector and r is the correlation coefficient calculated earlier. The t-value can be negative as a consequence of the formula, but the absolute should be taken when calculating the p-value.

When calculating the p-value using the t-statistic we will have to make the call on whether to go for a 1 tailed or 2 tailed test. Since the context is to validate whether the Pearson coefficient is statistically significant, no matter whether in the positive or negative direction, we will go for a 2 tailed hypothesis test.

Using this t value to find our p value from the t-stat table (using 2 tailed hypothesis test), we get this:

T VALUE	**-1.482**
P VALUE	**0.169**

In our code t-value applies the above mentioned formula while p-value uses the pt function to return the cumulative density function of the t-distribution using the t-value and degrees of freedom (number of datapoints) as its arguments

b) Code:

```
t_value = (r * sqrt(N-2)) / sqrt(1 - (r^2))
p_value = 2 * pt(-abs(t_value), N)
Output:
```

t-value table

	v1	v2	v3	v4	v5	v6
v1	Inf	-1.48194736	-0.44316524	-1.5481578	-1.85650876	1.5630238
v2	-1.4819474	Inf	0.03702242	1.8667101	1.02824364	-0.6109710
v3	-0.4431652	0.03702242	Inf	-0.8924306	0.06703315	-0.1047866
v4	-1.5481578	1.86671006	-0.89243058	Inf	0.86052595	-0.1749150
v5	-1.8565088	1.02824364	0.06703315	0.8605259	Inf	0.4369947
v6	1.5630238	-0.61097096	-0.10478658	-0.1749150	0.43699473	Inf

p-value table

	v1	v2	v3	v4	v5	v6
v1	0.00000000	0.16916501	0.6670796	0.15262458	0.09304656	0.1491118
v2	0.16916501	0.00000000	0.9711958	0.09150123	0.32806403	0.5548513
v3	0.66707961	0.97119577	0.0000000	0.39313071	0.94787661	0.9186171
v4	0.15262458	0.09150123	0.3931307	0.00000000	0.40965652	0.8646372
v5	0.09304656	0.32806403	0.9478766	0.40965652	0.00000000	0.6714008
v6	0.14911176	0.55485126	0.9186171	0.86463720	0.67140080	0.0000000

c and d) Explanation and Critical Analysis of Results:

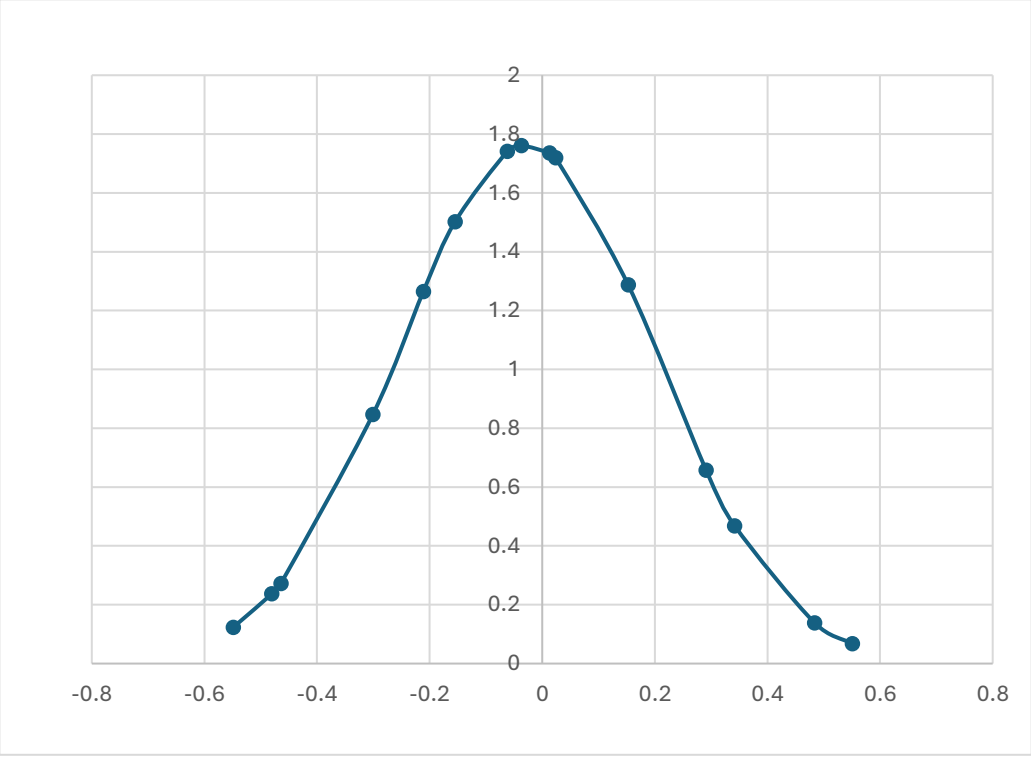
- Like our Pearson Coefficient table, our t and p value tables contain 15 unique values (when excluding the repeat entries and self-vector correlations)
- Out of these 15 p-values, there are 2 that are statistically significant: ***v1 & v5*** and ***v2 & v4*** these were the only 2 entries with a correlation coefficient greater than 0.5/less than -0.5.
- As these t and p-values are derived from the Pearson coefficient it is important to analyse that dataset further to gain a better understanding on why supposed random values are found to be statistically significant with each other

Sub-Challenge 4] Interpret the results and discuss any correlations that are statistically significant (p - value less than 0.10). Discuss the importance of p-values in determining the significance of correlations. Do you get any statistically significant correlations? If so, how can you justify it given that the variables are random and should not have any correlation?

- As mentioned above, v1&v5 and v2&v4 were found to be statistically significant using p-value < 0.1 as our indicator.
- v2&v4 were discussed earlier as they both had a rare output >2 in their 6^{th} row leading to a high coefficient.

- v1&v5 had a statistically significant negative Pearson correlation due to v5 and v1 typically printing values of opposite signs for their respective rows (if v1 outputted a + sign, v5 would coincidentally usually output a –‘ve sign value)
- These correlations and low pvalues occurred by chance but the computing and formulas interpreted these as statistically significant.
- One of the factors is the low number of datapoints for each vector, since we are only comparing 10 datapoints for each vector, there was a decent chance that a few of these 15 correlation coefficients would be statistically significant. Having a larger number of datapoints would reduce the odds that randomly generated numbers would coincidentally show signs of correlating with one another consistently.
- Another factor was discussed initially in this assignment (refer sub challenge 1), that the random numbers don’t all occur with equal probability but follow the probability density of a normal distribution curve, where values between -1 to +1 have a much higher chance of occurring than data that is greater than 1 or less than -1. This weighted probability along with the symmetry that a normal distribution provides factors into these correlations occurring as similar values b/w -1 to +1 are more likely to be repeated in datapoints between vectors. This is also supported by the output of the coefficients as we have 8 unique negative R values and 7 positive R values - a relatively even split.
- The above point is further supported when plotting the R values and their probability density functions.

x	y
-0.54873	0.123442
-0.48014	0.237144
-0.4641	0.272611
-0.3009	0.846589
-0.21114	1.265008
-0.15479	1.502082
-0.06172	1.741689
-0.03702	1.760757
0.013088	1.735413
0.023693	1.719259
0.152689	1.287085
0.291069	0.65751
0.341662	0.468519
0.483673	0.138575
0.550832	0.067913
MEAN	-0.02679
STD DEV	0.226343



- Here we can see that even with the limited 15 datapoints, we achieve a bell shaped curve – like that of the normal distribution curve.

- Another factor is the p-value boundary we set to determine if the coefficient was statistically significant. For weighted random values with a low number of datapoints, having a high p-value boundary such as 0.1 can lead to the computation determining that some entries are statistically significant when they are caused by chance.

- While p-values are a great tool to determine statistical significance, it is important to choose an appropriate boundary value given the context of the problem. It is also important to not take that p-value boundary as a hard truth. The p-value is a suggestion, given with a certain amount of confidence from the program, it is up to the individual to determine that the p-value makes sense for the data and the context of the problem.