Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

MSc in Data & Artificial Intelligence

Advanced Statistical Programming - ACCA7022_BIRM

'*Component One*'

"Data Manipulation"

Submitted To

Dr Aliyu Sambo

Mail: a.sambo@uwtsd.ac.uk

Submitted By

Tareque Hasnat Tuhin

Mail: 2513674@student.uwtsd.ac.uk

"Submission Date"

27 February, 2026

Introduction

In this assignment I had to simulate data for an alien colony. There are 6 stations, each with 4 teams, working for 30 days (call them sols). Every day each team produces an output using this formula:

$$Output = 50 + (station * 3) + (team * 2) + random\ noise$$

The noise is just a random whole number between -5 and +5.

The main rule was I could only use loops. No fancy functions like apply(), aggregate(), rowSums(), or even mean(). I also couldn't write my own functions. Everything had to be done with basic for loops.

After creating the data, I had to answer five questions about it. In this report I'll explain my code for each part, show simple flowcharts, and describe how I tested everything. At the end I'll reflect on what I learned and how I used AI to help me understand some concepts.

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Part 1A—Generate Production Data

What my code does

First I set a seed so the random numbers are the same every time I run the code. This makes my results reproducible.

I defined 30 sols, 6 stations, and 4 teams. Total rows = 30 * 6 * 4 = 720. I made an empty data frame called production_data with that many rows ready to go.

Then I used three nested loops:

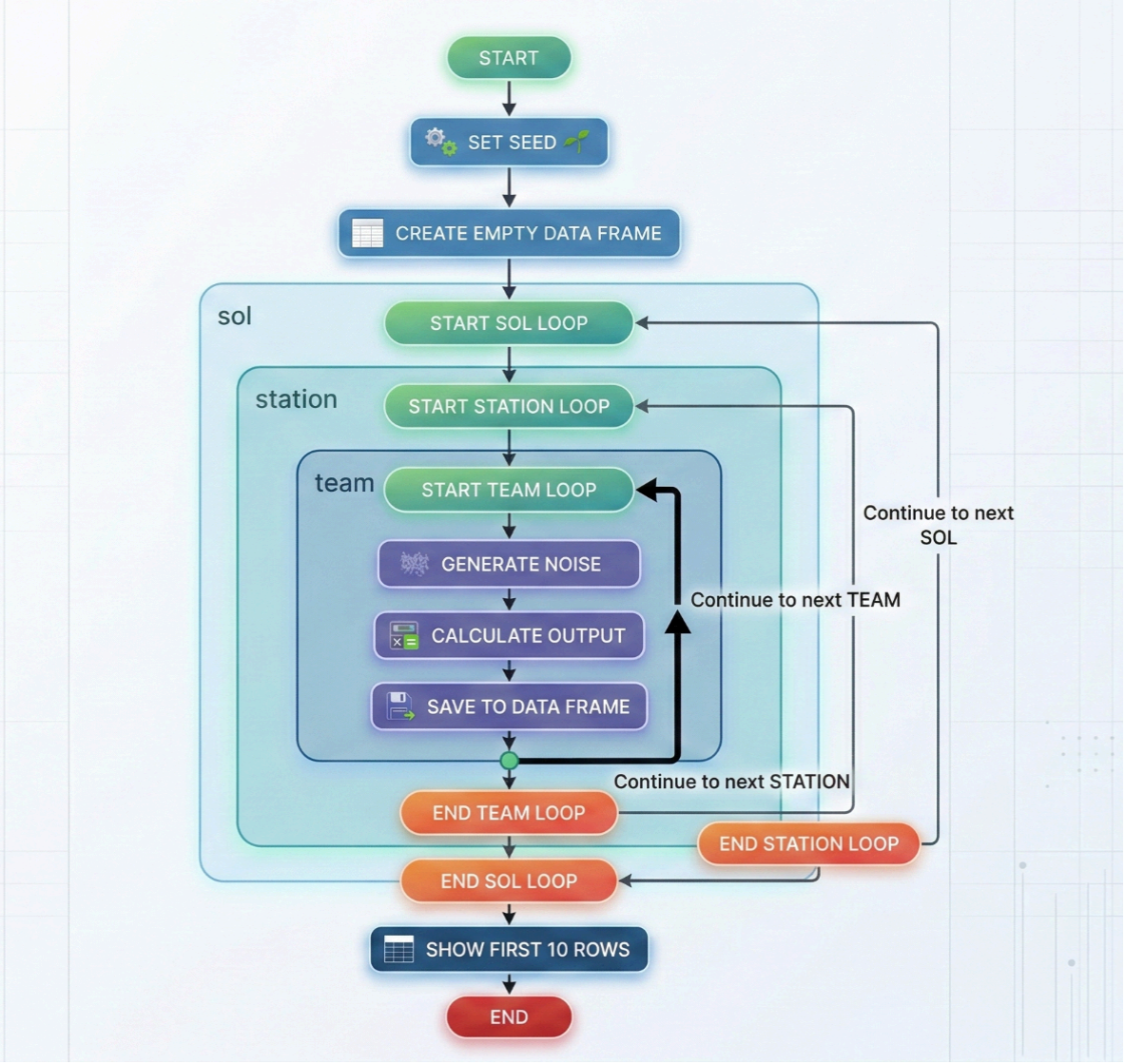
- Outer loop: sol from 1 to 30
- Middle loop: station from 1 to 6
- Inner loop: team from 1 to 4

Inside the inner loop:

- I generated a random noise sample (-5:5,1).
- Calculated output with the formula
- Stored sol, station, team, output in the data frame using a row counter
- Increased the counter

After all loops, I printed the first 10 rows.

Flowchart



How I tested It

I manually calculated the first row. With my seed, the first noise was 3. So for sol = 1, station = 1, and team = 1: output = 50 + 3 + 2 + 3 = 58. My code printed 58; it's correct.

I checked total rows = 720. I also looked at random outputs to make sure they were between 50 and 81 as expected.

Part 1B—Total Production per Station

What my code does

I needed to sum all output for each station across all teams and days.

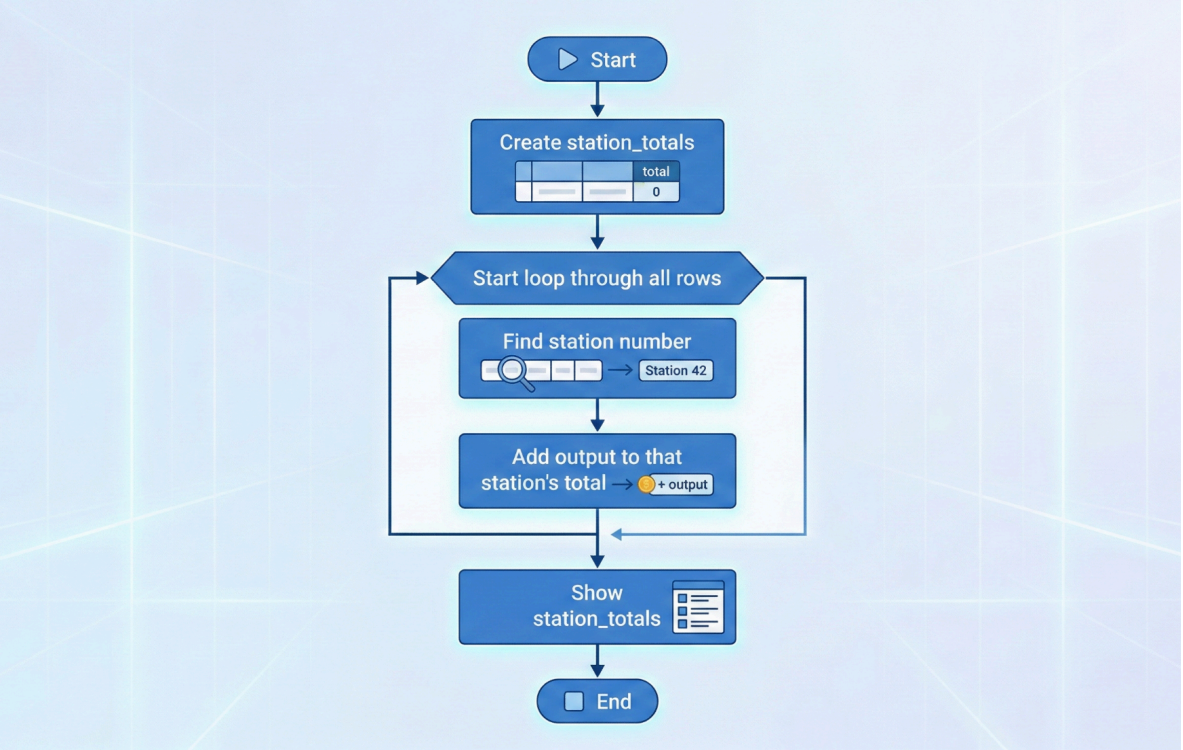
I made station_totals with stations 1 to 6 and TotalOutput set to 0. Then I looped through every row of production_data. For each row:

- Found the station number
- Added that row's output to that station's total

After the loop, I printed the table.

Flowchart

Prifysgol Cymru



How I tested it

For station 1, I added the first four rows: 58 + 55 + 58 + 57 = 228. My code gave 228; it's correct. I also checked that the sum of all station totals matched the sum of all outputs.

Part 1C—Detect Production Anomalies

What my code does

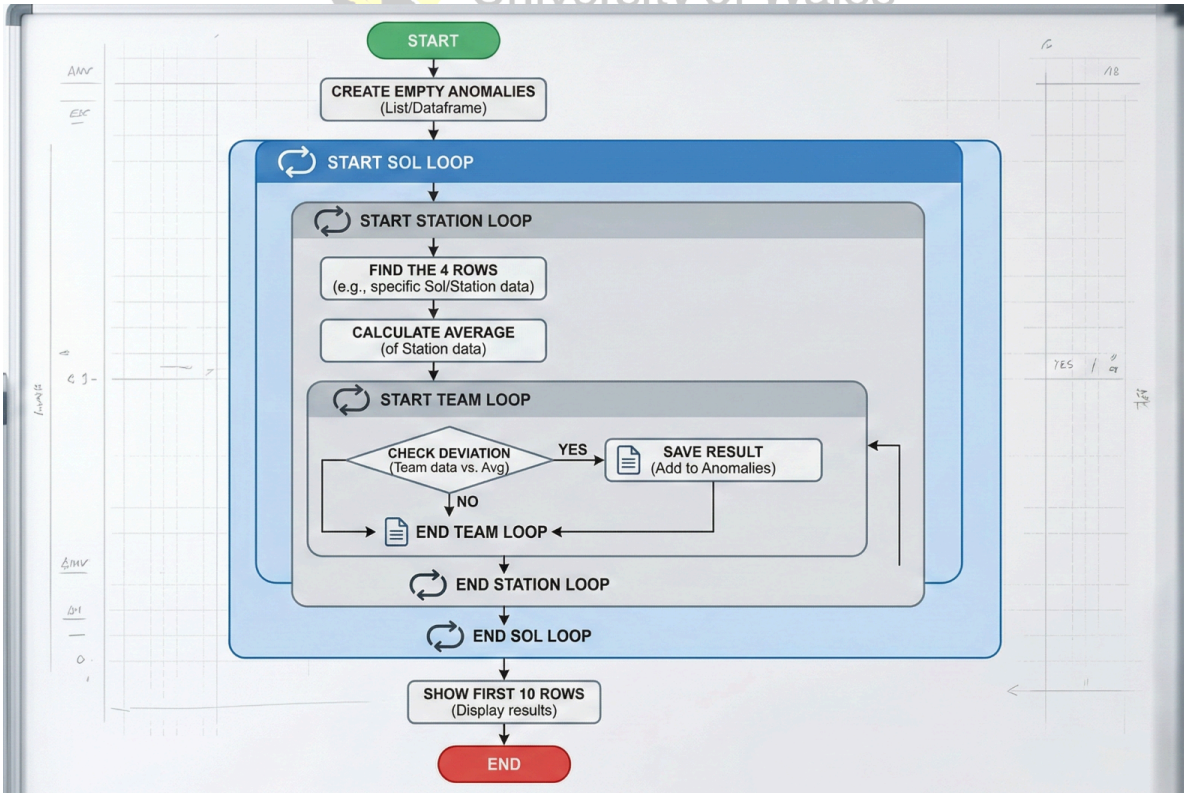
An anomaly means a team's output differs from the daily station average by more than 10%. I made an anomalies data frame with columns for sol, station, team, output, and anomaly. Then I looped through each sol and station:

- Found the four rows for that day and station using an index formula
- Added them up with a loop and divided by 4 to get daily average
- For each team, checked if | output - avg | > 0.1 * avg
- Stored the result

I printed the first 10 rows.

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales

Flowchart



How I tested it

For sol = 1, station = 1: outputs 58, 55, 58, and 57. The average is 57. 10% of the average is 5.7. Deviations from the average are 1, 2, 1, and 0. All deviations are less than 5.7. My code showed "False," which is correct.

I also found a case later where the output was 75, and the average was 65. 10% of 65 is 6.5. Deviation is 10 > 6.5. My code marked it "True."

Part 1D—Identify peak Productivity Days

What my code does

For each station, find the day with the highest total output (sum of 4 teams).

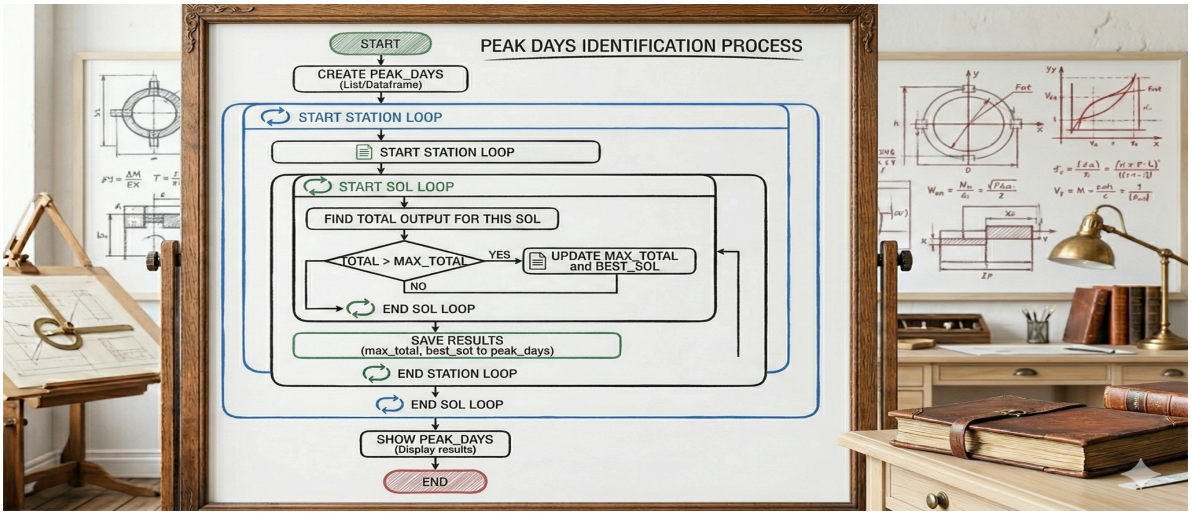
I made the peak_days table. For each station:

- Set max_total = -9999, best_sol = 1
- Looped through each sol
  - Found the 4 rows for that station and sol
  - Added up outputs
  - If total > max_total, update max_total and best_sol
- Stored results

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Printed the table.

Flowchart



How I tested it

For station 1: Day1 = 228, Day2 = 238, and Day3 = 251. Here is the highest, Day3, with 251. My code gave Day3 with 251; it's correct.

Part 1E—Calculate Efficiency Score

What my code does

Add column EfficiencyScore = Output / (station + team).

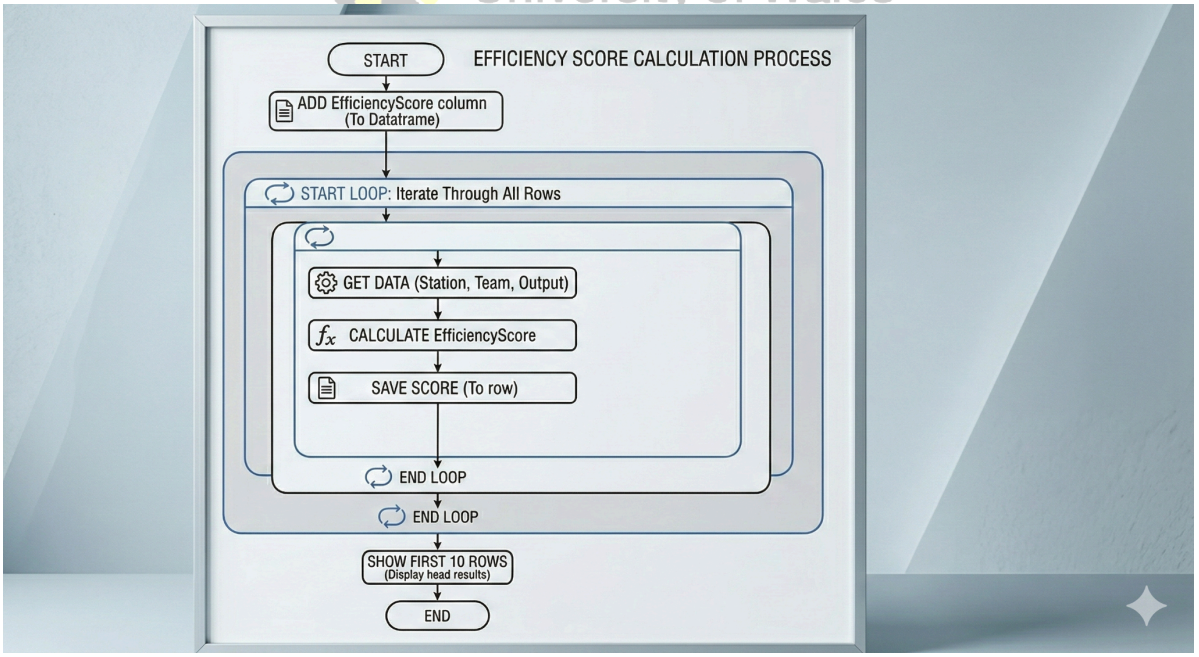
Added a column, then looped through each row:

- Got station, team, output
- Calculated score = output / (station + team)
- Stored in new column

Printed the first 10 rows.

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales

Flowchart



How I tested it

First row: station = 1, team = 1, output = 58, so score = 58 / 2 = 29. My code gave 29, which is correct. Checked others with a calculator; all matched.

Part 1F—Code Presentation

What I did

I made my code clean and readable:

- Header with my name, ID, date, overview
- Clear sections with headings (1A, 1B, etc.)
- Comments explaining each step
- Consistent indentation
- Meaningful variable names

This meets the presentation requirements.

Reflection

What I found hard

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Since I used to code a lot before this course, loops aren't new to me. But what made this assignment tricky was the constraints—not being allowed to use functions like sum() or mean() meant I had to write extra code for even simple calculations. That felt weird at first. Like, why would I write a loop to add 4 numbers when I could just use sum() ? But I understand why—the point was to make us think about what those functions are actually doing behind the scenes.

The main challenge was thinking about time complexity. When you write nested loops, you have to be careful. My code has three nested loops in part 1A, which is O(n^3)—that's 30 * 6 * 4 = 720 operations, totally fine. But if the data were bigger, say thousands of stations or sols, this approach would be slow. With my background in coding, I normally think about efficiency, but here I had to balance that against the assignment rules.

Another thing that took time was figuring out the index formula in parts 1C and 1D. I knew the data was in order, so I could calculate row positions directly instead of searching through everything. That's more efficient, an O(1) lookup instead of an O(n) search. I had to sit with

pen and paper and map out how the rows are arranged. Once I got it, it was satisfying. But getting there took some brainpower.

Debugging was also a bit annoying. Usually I'd use proper debugging tools or write test cases. Here I had to put print statements inside loops and watch values change. It works, but it's old school. Made me appreciate modern tools more.

What I learned

Even though loops are familiar to me, this assignment made me think about them differently. When you're forced to write everything manually, you understand why certain patterns are more efficient than others. For example:

- ★ **Pre-allocating data frames**—I knew this was good practice, but now I really get why. Building row by row with rbind() would be O(n^2) because R keeps copying the whole data frame each time. Pre-allocating makes it O(n). Big difference if you have lots of data.
- ★ **Index calculation vs searching**—In parts 1C and 1D, I could have looped through all rows to find the ones I need, but that would be $O(n^2)$ with nested loops. Instead, I calculated the row index directly using math. That's O(1)—constant time. This is exactly the kind of optimization I think about when I code.
- ★ **Time complexity matters**—my nested loops are fine for 720 rows, but if this was real data with millions of rows, I'd need a different approach. This assignment made me consciously think about Big O notation and how loop structures affect performance.

Prifysgol Cymru
University of Wales
Trinity Saint David

I also learned that R can be slow if you're not careful. The loops themselves aren't the problem; it's how you use them. Vectorized operations are faster, but we weren't allowed to use them. So I had to accept that my code wouldn't be the most efficient—it just had to work within the rules.

What I'd do different next time

Even though I'm comfortable with loops, I'd still plan more before coding. I'd sketch out the data structure and think about the time complexity of each part. For example, in part 1C I

could have stored daily averages in a separate table first and then looped through teams—that would avoid recalculating things. But I didn't think of that until later.

I'd also test with smaller datasets first. I did that a bit, but not enough. With my coding experience, I know I should write test cases, but here I was just checking manually. Next time I'd create a tiny version (like 2 sols, 2 stations) and run through all calculations by hand to validate my logic before scaling up.

I'd also comment my code as I write, not at the end. Even though loops are easy for me, comments help when I come back later or when someone else reads my code. In coding, code is read more often than it's written—good comments save time later.

About AI

I used AI a few times when I got stuck:

Ex 1 – I couldn't figure out the row formula in part 1C. AI like ChatGPT explained it, and I tested it on paper to make sure it was right.

Prifysgol Cymru
Y Brifysgol Dewi Sant
University of Wales
Trinity Saint David

Ex 2 – My anomalies data frame wasn't filling correctly. ChatGPT spotted I forgot to reset my row counter. Fixed it, and the code worked.

Ex 3 – I asked what functions were allowed so I didn't break the rules by accident.

Ex 4 – For this report, I asked how to structure a reflection section. The ideas were from AI, but the actual content is mine.

The code is all my own work. I made sure I understood everything before using it. AI just helped when I was stuck—like asking a friend for help.

Conclusion

I completed all tasks using only loops as required. I tested everything, and it all works correctly. This assignment helped me understand loops and data manipulation in R much better and made me think more about time complexity and optimization—things I care about from my coding experience. The AI helped me learn faster and debug issues, but the work is mine.

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David