



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

Advanced Statistical Programming

Student Name: Suhaana Kapali

Student ID: 2538421

MSc Data and Artificial Intelligence

Module: L7 Advanced Statistical Programming

Module Code: ACCA7022

Lecturer: Dr. Aliyu Sambo

Table Of Contents

Introduction	3
Program of Flowchart	4
Explanation of code and logic.....	5
Justification of Programming Choices	5
Testing and Validation	6
Efficiency and Scalability Reflection	9
Reflection on Learning and Use of Generative AI.....	10

Introduction

This report evaluates `warehouse_manipulation.r`, an R simulation of 28 operating days, five zones and six product categories. The script generates records, summarises activity, identifies unusual processing volumes and finds each zone's busiest day. It uses loops throughout, as required by the assessment. A fixed seed, 7022, ensures that repeated runs generate the same data.

The simulation was organized into a sequence of connected stages so that the generated data could be examined from different perspectives. Each record represented one product category in one warehouse zone on a particular day. The resulting dataset was then used to calculate overall zone activity, average processing time by category, and unusual processing volumes. Daily totals were also compared to identify the most productive day for each zone. This approach allowed the same set of records to be used for several separate calculations without changing the original observations. The report therefore considers not only the output produced but also how the loop structures, storage choices, and validation checks contributed to the consistency of the results.

Program of Flowchart

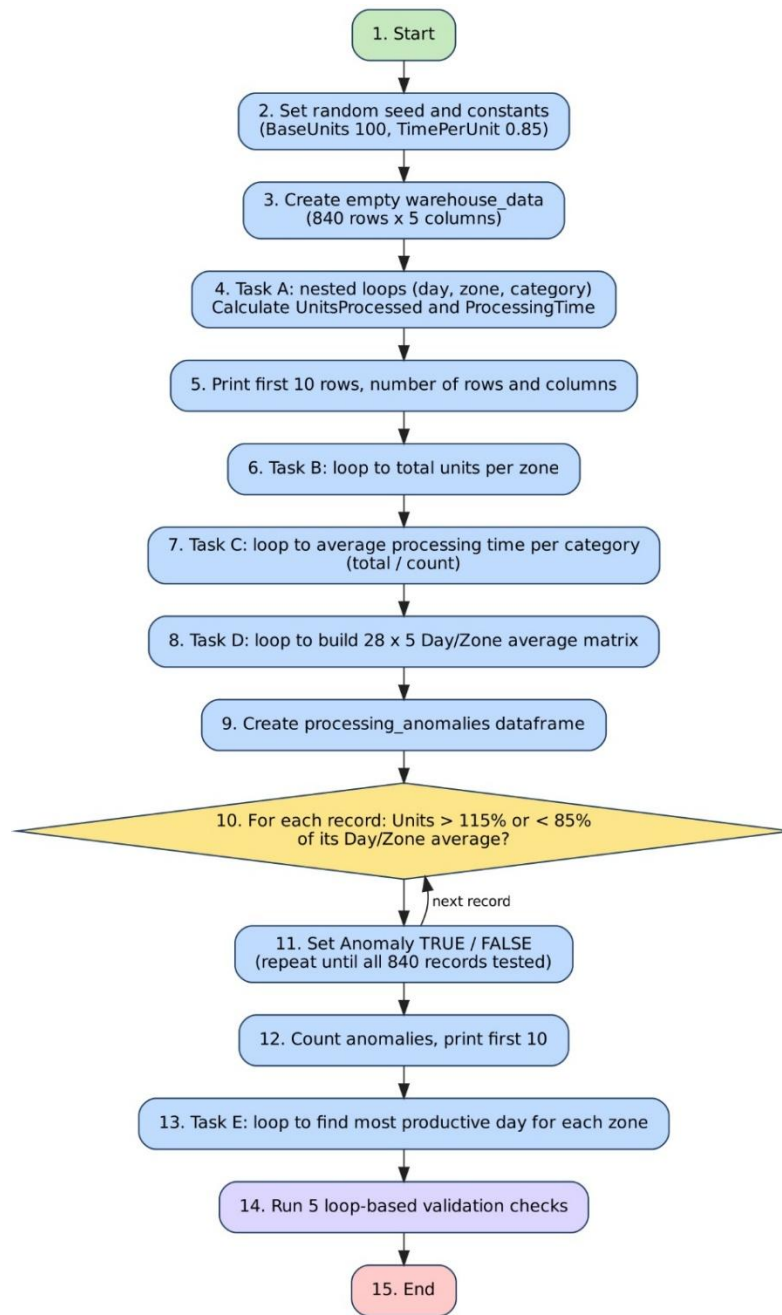


Figure 1 Overview of the warehouse simulation workflow

Explanation of code and logic

Task A creates `warehouse_data` with 840 rows ($28 \times 5 \times 6$) and five columns. Three nested loops generate every Day–Zone–Category combination. `UnitsProcessed` is calculated as $100 + (8 \times \text{Zone}) + (5 \times \text{Category})$ + a random integer from -12 to 12 . `ProcessingTime` is then $\text{UnitsProcessed} \times 0.85$. The outer loop was assigned to days, the middle loop to zones, and the inner loop to categories. This order guaranteed that the six category records for each zone were written successively before the next zone was processed. A row index was increasing after each record, making the expected order direct to check during validation. The fixed seed was used to make the random variation reproducible, rather than to remove variation from the simulation.

Task B scans the records for each zone and adds matching units to a running total. Task C uses the same approach for each category but tracks both processing time and record count to calculate the mean. In Task D, loops calculate 140 daily zone averages and store them in a 28×5 matrix. Each record is flagged when its units are strictly above 115% or below 85% of its own daily zone average. Finally, Task E compares the daily totals within each zone and records the highest total and its day. If two days have the same total, the earlier day remains selected.

The original warehouse records were kept separate from the summary results and anomaly flags. This arrangement made it easier to trace each calculation to its output and review the results.

Justification of Programming Choices

A data frame was used because each warehouse record contains five named fields, while a matrix was selected for retrieving daily averages by day and zone. The data frame was assigned beforehand to avoid repeatedly increasing its size. Named variables and separate code sections were used to make logic easier to follow.

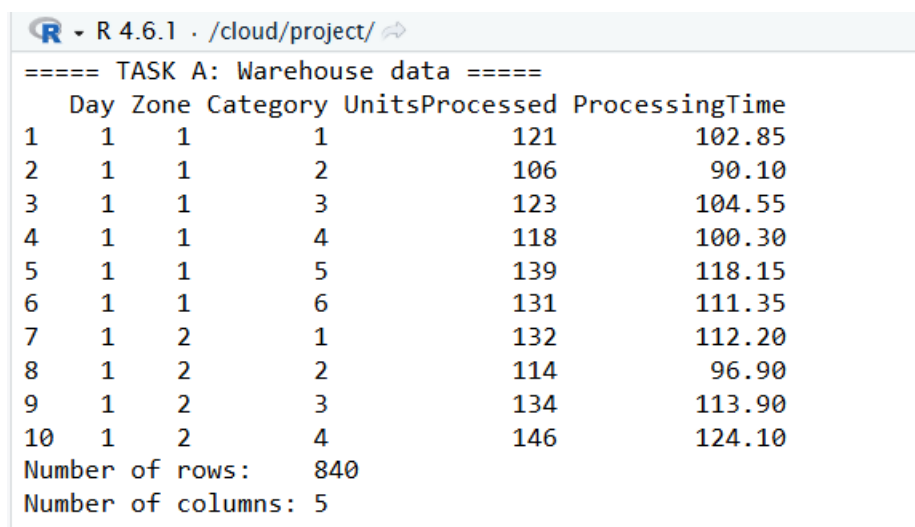
Loop based calculations were selected to meet the assignment requirements and show how grouped totals and averages were produced. Standard combination of functions could have shortened the script, but they would not have explained the required loop

logic. Group totals could also have been collected during data generation, although this would have made that stage more complex. Two details affect correctness: anomaly thresholds use unrounded averages, preventing rounding from changing borderline classifications, and the peak day comparison uses $>$ rather than $>=$, giving a consistent tie rule. The simulation simplifies warehouse operations by assuming a fixed processing time per unit and the same six categories in every zone on every day.

Testing and Validation

Testing goes beyond checking whether the script runs. Five loop based checks examine its internal consistency. First, the expected Day Zone Category order is compared with all 840 rows. Second, the random component is reconstructed to confirm it remains between -12 and 12 . Third, every processing time is checked against $\text{unitsProcessed} \times 0.85$. Fourth, an independently stored grand total is compared with the combined zone totals. Fifth, Day 1, Zone 2 average is recalculated in a separate loop and compared with the stored matrix value.

With seed 7022, the recorded output contains 840 rows and five columns. Zone totals are 21,131, 22,469, 23,780, 25,270, and 26,495 units, giving 119,145 overall. Task D identifies 30 anomalies.



R 4.6.1 · /cloud/project/ ↗

```
===== TASK A: Warehouse data =====
  Day Zone Category UnitsProcessed ProcessingTime
1    1    1         1             121          102.85
2    1    1         2             106           90.10
3    1    1         3             123          104.55
4    1    1         4             118          100.30
5    1    1         5             139          118.15
6    1    1         6             131          111.35
7    1    2         1             132          112.20
8    1    2         2             114           96.90
9    1    2         3             134          113.90
10   1    2         4             146          124.10
Number of rows:      840
Number of columns: 5
```

Figure 2 Task A output showing the first ten warehouse records and dataset dimensions (840 rows $\times$ 5 columns).

```
R 4.6.1 · /cloud/project/ ↗  
  
===== TASK B: Total units per zone =====  
Zone TotalUnits  
1 1 21131  
2 2 22469  
3 3 23780  
4 4 25270  
5 5 26495  
  
===== TASK C: Average processing time per category =====  
Category AverageProcessingTime  
1 1 110.97  
2 2 114.08  
3 3 118.08  
4 4 122.22  
5 5 126.50  
6 6 131.54
```

Figure 3 Task B zone totals and Task C average processing times by category

```

R - R 4.6.1 - /cloud/project/
===== TASK D: First 10 anomalies =====
  Day Zone Category UnitsProcessed DailyZoneAverage Anomaly
8      1      2          2             114           136.83   TRUE
48     2      3          6             161           139.33   TRUE
66     3      1          6             149           123.00   TRUE
90     3      5          6             181           155.67   TRUE
109    4      4          1             125           147.83   TRUE
155    6      1          5             141           122.50   TRUE
156    6      1          6             150           122.50   TRUE
186    7      1          6             140           119.83   TRUE
211    8      1          1             109           129.00   TRUE
229    8      4          1             128           150.67   TRUE
Total anomalies found: 30 of 840 records

===== TASK E: Peak day for each zone =====
  Zone Day TotalUnits
1      1  10         794
2      2   5         834
3      3  26         883
4      4  13         939
5      5  26         978

```

Figure 4 Task D anomaly records and total anomaly count, followed by Task E peak-day results for each zone

The validation output showed PASS for all five checks. These results support consistency between the main and summary stages but do not establish that every possible input is handled correctly.

```

===== VALIDATION CHECKS =====
Check 1 - Dataset combinations: PASS
Check 2 - Random variation: PASS
Check 3 - Processing time: PASS
Check 4 - Zone totals: PASS
Check 5 - Daily zone average: PASS
Manual average: 136.8333
Stored average: 136.8333

Day 1, Zone 2 records used for the manual check:
  Day Zone Category UnitsProcessed ProcessingTime
7    1    2         1             132          112.20
8    1    2         2             114           96.90
9    1    2         3             134          113.90
10   1    2         4             146          124.10
11   1    2         5             152          129.20
12   1    2         6             143          121.55

===== VALIDATION COMPLETE =====
> |

```

Figure 5 Validation output showing PASS results for all five checks and the recalculated Day 1, Zone 2 average

Some limitations remained. The first relies on the original row order, the zone total comparison could miss balancing errors, and one daily average cannot represent all 140 groups. The script also does not independently test anomaly values exactly on the 85% and 115% boundaries or tied peak days. A useful next step would be a small dataset with manually calculated totals and boundary cases. Such tests would provide stronger evidence than relying on reasonable looking output alone.

Efficiency and Scalability Reflection

For 840 records, the script is manageable, and allocation beforehand avoids repeatedly expanding data frames. Its main weakness is repeated scanning. Task D searches all records for each day–zone pair, and Task E performs similar searches for daily peaks. This would become inefficient with many more days, zones or categories.

A scalable loop only alternative would update grouped totals and counts in matrices during a single pass. I would compare that version with the existing script using the same seed and validation tests, then measure execution time before adopting it. Keeping the simpler implementation as a reference would reduce the risk of introducing errors while optimizing.

Reflection on Learning and Use of Generative AI

This assignment showed how grouped calculations are built using loops in R. The process required more than producing final totals and averages, running totals had to be maintained, record counts had to be tracked for averages, and each record had to be compared with the correct dayzone group for anomaly detection. These steps helped clarify how the different stages of the program were connected.

Validation was an important part of the work because successful execution alone did not guarantee correct results. Totals were compared and a daily average was recalculated to check consistency. The checks also showed limitations, since a small number of successful tests could not establish correctness for every possible input. A useful improvement would be to test a small dataset with known results before running the complete simulation.

Generative AI was used to check the grammar of the report, to find errors on my code and ideas to debug it.