International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Impact of Different Data Management Frameworks on Common Data Management Tasks in Information System (R Language Perspective)

Anant Prakash Awasthi[1], Niraj Kumar Singh[2], Masood H Siddiqui[3], Aanchal A Awasthi[4]

Submitted: 04/02/2024 **Revised**: 13/03/2024 **Accepted**: 17/03/2024

Abstract: To maximize data processing and analysis, effective data management is essential. It ensures that data is efficiently processed, readily accessible, secure, and well-organized. This enhances data integrity, reduces the amount of redundancy, and it makes decision-making more prompt. In an era where data is a valued asset that drives innovation and strategic decision-making, effective data management techniques are essential.

The two essential data management activities for improving data processing are joining and sorting. By combining datasets based on common characteristics, joining makes thorough analysis easier. Sorting data well enhances search and retrieval. When combined, these processes enhance the accuracy and speed of data processing, simplifying workflows and enabling sound decision-making. Database management systems depend on joining and sorting to enable the creation of value, the extraction of significant insights, and the identification of trends from massive datasets.

The performance of native R, tidyverse, and data.table when merging data in R varies. Large datasets may cause Native R to lag, despite its versatility. Known for its readability, Tidyverse strikes a balance between performance and simplicity. Because of its exceptional speed, Data.table is a very effective option for large-scale data joins. The decision is based on the complexity and amount of the dataset. The best option for maximum performance, particularly for complex and large-scale jobs, is Data.table. Native R and Tidyverse work well with smaller, more manageable datasets when code readability is crucial. Every method addresses particular requirements in R data analysis. Similarly, when it comes to sorting data in R, Native R, tidyverse, and data.table behave differently. While Native R provides a standard method, it might not be as effective with larger datasets. Although readability is given priority in Tidyverse's user-friendly syntax, it may not be as fast as more efficient options. Once more, Data.table runs faster and uses less memory when sorting large amounts of data than the competition. The decision is based on the needs of the analysis: data.table for best performance, especially with large datasets and computationally intensive tasks; tidyverse for readability; and Native R for simplicity.

Hence, in order to sum up, effective data management is essential for businesses to fully utilize their data and make wise decisions. Optimizing data processing and analysis requires careful consideration of joining, sorting, and tool selection.

Keywords*: Memory Management in R, Performance in R, Native R, Tidyverse, Data.Table*

1. Introduction:

In the contemporary landscape of information systems, the significance of data as a pivotal asset for driving innovation and strategic decision-making cannot be overstated. Effective data management serves as the linchpin in maximizing the processing and analysis of data, ensuring its efficiency, accessibility, security, and organization. This imperative practice not only enhances data integrity but also streamlines workflows, reduces redundancy, and facilitates prompt decision-making.

Two fundamental data management activities, namely joining and sorting, play a crucial role in improving data processing capabilities. Joining, by amalgamating datasets based on common characteristics, simplifies thorough analysis, while sorting data optimally enhances search and retrieval processes. The synergy of these activities contributes to the accuracy and speed of data processing, empowering information systems to extract meaningful insights and identify trends from vast datasets.

In the realm of data management within the R language, the performance of different frameworks – namely Native R, Tidyverse, and Data.table – significantly influences the outcome of data processing tasks. The choice among these frameworks hinges on the complexity and scale of the dataset at hand. Native R, known for its versatility, may experience lag with large datasets. Tidyverse, renowned for its readability, strikes a balance between performance and simplicity, making it suitable for smaller, more manageable datasets. On the other hand, Data.table emerges as a powerful option for large-scale data operations, excelling in speed and efficiency.

This paper delves into the impact of these data management frameworks on common data management tasks within the R language, focusing on joining and sorting operations. The assessment considers the

[1, 2, 4] *Department of Statistics, Amity Institute of Applied Sciences, Amity University, Noida, Uttar Pradesh*
[3] *Department of Statistics, University of Lucknow, Lucknow, Uttar Pradesh*
anant.awasthi@outlook.com
nksingh5@gmail.com
mhsiddiqui@gmail.com
draanchalawasthi@gmail.com

performance variations among Native R, Tidyverse, and Data.table, providing insights into their strengths and limitations in handling datasets of different complexities and sizes. By understanding the nuances of these frameworks, researchers, practitioners, and data analysts can make informed decisions in selecting the most appropriate tool for their specific data analysis needs.

As businesses increasingly recognize the pivotal role of effective data management in harnessing the full potential of their data, this research seeks to contribute valuable insights into the optimization of data processing and analysis. The paper aims to guide practitioners in making informed decisions regarding joining, sorting, and tool selection, emphasizing the pivotal role these aspects play in achieving optimal performance and extracting meaningful insights from diverse datasets.

2. Frameworks (Packages) in Scope:

a. Native R (utils)[1];
b. tidyverse_1.3.2/tibble_3.1.8[2];
c. data.table_1.14.2 [3]

3. Methodology

There was no significant work was found in the literature for benchmarking of data management tasks using R frameworks. We have discussed here approach to record the memory utilization and execution time of frameworks along with analysis approach which includes graphical exploration to establish the benchmark and statistical inference to support the evidences.

An initial sample of 10^2 records has been drawn randomly with replacement from flight data from nycflight13 package and changed the data structure to data structure in scope. Execution time was measured and results were written to a file. In the next step, the sample size has been stepped up by a multiple of 10^1 rows and execution time was recorded for each task and frameworks in scope. The sample size was increased to 10^5 records.

Fig 3.1 – Experiment Control Workflow

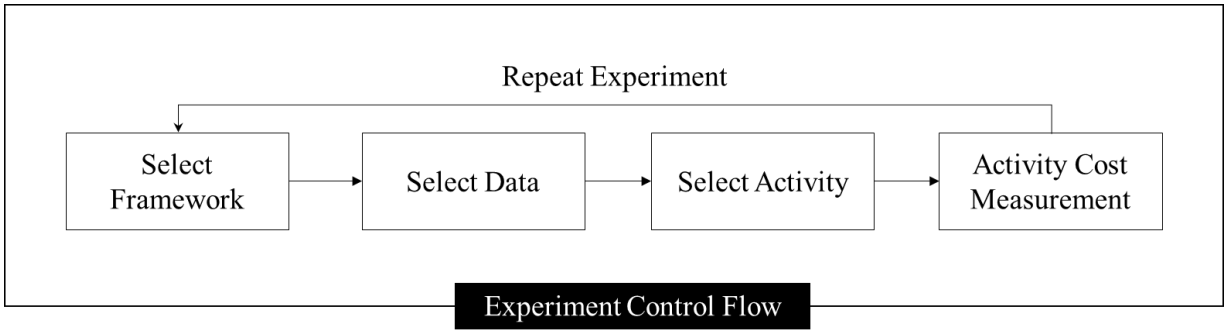
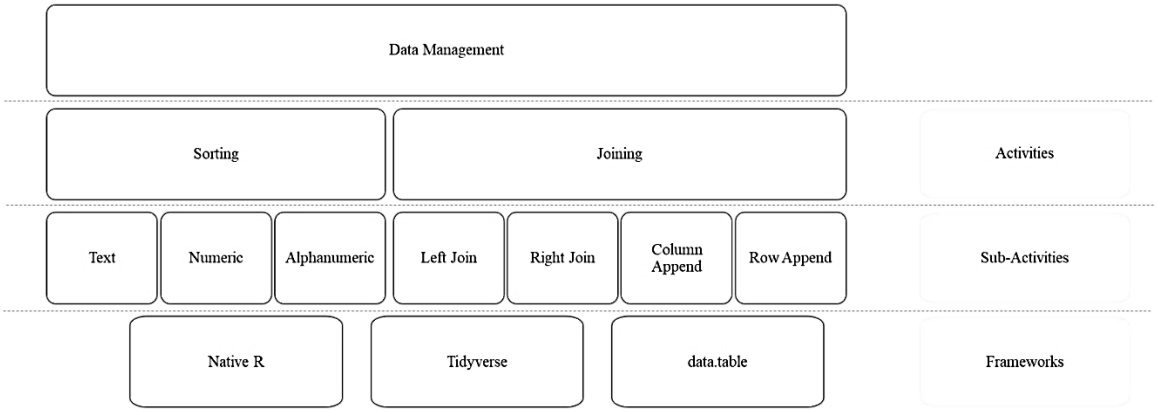


Fig 3.2 – Data management activities and sub-activities



Each dataset was processed for each task with different frameworks and cost value (execution time) has been recorded. The environment get cleaned after each iteration to avoid any garbage in the environment. Each activity gets repeated 50 times to avoid any bias in the execution cycle.

4. Methods:

Exploratory Data Analysis (EDA) is a crucial step in the data analysis process that involves the initial examination and exploration of a dataset to gain insights, detect patterns, and identify potential trends or outliers. It serves as the foundation for more in-depth analysis and helps researchers and analysts understand the underlying structure of the data.

An exploratory data analysis has been performed on machine data for each activity, sub-activity and framework. Numeric exploratory data analysis was performed to get an overview of performance of data management frameworks. Later the same has been

established with Graphical exploratory data analysis to confirm, there is no misleading summary due to data distribution.

Analysis of Variance (ANOVA) is a statistical technique used to analyze the mean difference of memory utilization between the groups (Frameworks). ANOVA allows to determine whether the means of two or more groups are significantly different from each other. Analysis of variance is used to establish the fact whether three data structure in scope have significant difference in memory utilization or not for at least one set of frameworks.

Analysis of variance has been performed with null hypothesis that there is no significant difference in memory utilization of three frameworks (Native R, Tidyverse and data.table).

H0: There is no significant difference in memory utilization of three frameworks (Native R, Tidyverse and data.table)

5. Analysis and Results

5.1 Analysis of frameworks performance in sorting task:

The analysis was performed on machine data collected for sorting task and sub-tasks defined in figure 3.2

5.1.1 Sorting of Numeric data and one variable task:

Table 5.1.1.1 *– Mean execution time for sorting numeric data with one variable in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0001092301	0.0002789582	0.0026437109
1,000	0.0001226501	0.0005052928	0.0031999175
10,000	0.0002534025	0.0023109786	0.0038007622
100,000	0.0007437662	0.0552861505	0.0223498222
1,000,000	0.0070738811	0.5270177412	0.4108667845

Table 5.1.1.2 *– Performance comparison sorting numeric data with one variable in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	39.16%	100.00%	947.71%
1,000	24.27%	100.00%	633.28%
10,000	10.97%	100.00%	164.47%
100,000	1.35%	100.00%	40.43%
1,000,000	1.34%	100.00%	77.96%

Table 5.1.1.3 *– Analysis of variance of execution time for sorting numeric data with one variable task*

Sample Size	F Statistic	P-value
100	2847.198832	2.8763E-118
1,000	574.8956323	3.17595E-70
10,000	747.6547728	9.15968E-78
100,000	2206.374908	2.3209E-110
1,000,000	1816.748274	2.2284E-104

5.1.2 Sorting of Numeric data and Multiple variables task:

Table 5.1.2.1 *– Mean execution time for sorting numeric data with multiple variables in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0001362677	0.0003335794	0.0031702710

1,000	0.0001554895	0.0009022470	0.0039134508
10,000	0.0004225555	0.0023501227	0.0053115033
100,000	0.0013343456	0.0520894654	0.0222365700
1,000,000	0.0140660846	0.5260348940	0.4159666619

Table 5.1.2.2 – *Performance comparison sorting numeric data with Multiple variables in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	40.85%	100.00%	950.38%
1,000	17.23%	100.00%	433.74%
10,000	17.98%	100.00%	226.01%
100,000	2.56%	100.00%	42.69%
1,000,000	2.67%	100.00%	79.08%

Table 5.1.2.3 – *Analysis of variance of execution time for sorting numeric data with multiple variables task*

Sample Size	F Statistic	P-value
100	1024.195846	4.97383E-87
1,000	357.3077313	3.56683E-57
10,000	432.954885	2.44553E-62
100,000	1984.194987	4.3513E-107
1,000,000	1394.571983	2.60729E-96

5.1.3 Sorting of Text data and Single variable task:

Table 5.1.3.1 – *Mean execution time for sorting text data with Single variable in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0001048509	0.0004756379	0.0027897569
1,000	0.0001749017	0.0018483864	0.0055657809
10,000	0.0002171909	0.0230633976	0.0256044599
100,000	0.0007404266	0.3418253105	0.3145452813
1,000,000	0.0075406263	4.9952621417	4.9316663550

Table 5.1.3.2 – *Performance comparison sorting text data with one variable in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	22.04%	100.00%	586.53%
1,000	9.46%	100.00%	301.12%
10,000	0.94%	100.00%	111.02%
100,000	0.22%	100.00%	92.02%
1,000,000	0.15%	100.00%	98.73%

Table 5.1.3.3 – *Analysis of variance of execution time for sorting text data with one variable task*

Sample Size	F Statistic	P-value
100	582.3752824	1.36696E-70
1,000	399.7928371	3.5487E-60
10,000	1245.256716	6.91855E-93
100,000	3569.386919	2.546E-125
1,000,000	3260.476957	1.7154E-122

5.1.4 Sorting of Text data and Multiple variables task:

Table 5.1.4.1 – *Mean execution time for sorting text data with Multiple variables in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0001536516	0.0004777348	0.0033136688
1,000	0.0001671399	0.0024942584	0.0051134566
10,000	0.0003902598	0.0297835725	0.0332884222
100,000	0.0013080937	0.4007011749	0.3499668288
1,000,000	0.0128390679	5.5760959891	5.5528008486

Table 5.1.4.2 – *Performance comparison sorting text data with Multiple variables in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	32.16%	100.00%	693.62%
1,000	6.70%	100.00%	205.01%
10,000	1.31%	100.00%	111.77%
100,000	0.33%	100.00%	87.34%
1,000,000	0.23%	100.00%	99.58%

Table 5.1.4.3 – Analysis of variance of execution time for sorting text data with multiple variables task

Sample Size	F Statistic	P-value
100	509.8179751	7.55224E-67
1,000	950.8216765	8.03613E-85
10,000	607.8017308	8.34962E-72
100,000	1188.941418	1.71038E-91
1,000,000	3359.806345	1.9827E-123

5.1.5 Sorting of Alpha-numeric data and Single variable task:

Table 5.1.5.1 – *Mean execution time for sorting Alpha-numeric data with Single variable in seconds*

Sample Size	data.table	Native R	Tidyverse

100	0.0002320735	0.0008733590	0.0045260824
1,000	0.0006350751	0.0068407380	0.0096418822
10,000	0.0020761554	0.0642521353	0.0544743591
100,000	0.0033552404	0.7980587956	0.7888056717
1,000,000	0.0111766409	10.7279374556	10.7455592345

Table 5.1.5.2 – *Performance comparison sorting Alpha-numeric data with one variable in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	26.57%	100.00%	518.24%
1,000	9.28%	100.00%	140.95%
10,000	3.23%	100.00%	84.78%
100,000	0.42%	100.00%	98.84%
1,000,000	0.10%	100.00%	100.16%

Table 5.1.5.3 – *Analysis of variance of execution time for sorting alpha-numeric data with one variable task*

Sample Size	F Statistic	P-value
100	2470.784849	7.2952E-114
1,000	450.0662526	2.12669E-63
10,000	559.694351	1.81613E-69
100,000	1457.216578	1.2088E-97
1,000,000	5405.244592	2.3937E-138

5.1.6 Sorting of Alpha-numeric data and Multiple variables task:

Table 5.1.6.1 – *Mean execution time for sorting Alpha-numeric data with Multiple variables in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0002079651	0.0005610256	0.0050519032
1,000	0.0005848977	0.0079561373	0.0112699413
10,000	0.0020820469	0.0599394282	0.0637417024
100,000	0.0037512825	0.6977975037	0.6710197818
1,000,000	0.0136423213	9.1822695426	9.4916779269

Table 5.1.6.2 – *Performance comparison sorting Alpha-numeric data with Multiple variables in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	37.07%	100.00%	900.48%
1,000	7.35%	100.00%	141.65%
10,000	3.47%	100.00%	106.34%

100,000	0.54%	100.00%	96.16%
1,000,000	0.15%	100.00%	103.37%

Table 5.1.6.3 – *Analysis of variance of execution time for sorting alpha-numeric data with multiple variables task*

Sample Size	F Statistic	P-value
100	7588.372324	4.7152E-149
1,000	776.6319064	7.15938E-79
10,000	693.1273543	1.42946E-75
100,000	2019.354389	1.2526E-107
1,000,000	5442.264376	1.4592E-138

5.2 Analysis of frameworks performance in joining/merging task:

The analysis was performed on machine data collected for joining/merging task and sub-tasks defined in figure 3.2

5.2.1 Adding columns to Dataframe:

Table 5.2.1.1 – *Mean execution time for adding columns to dataframe in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0000002080	0.0000667021	0.0003022237
1,000	0.0000002369	0.0000597766	0.0003120869
10,000	0.0000001573	0.0000609473	0.0002889712
100,000	0.0000001714	0.0000581416	0.0002806376
1,000,000	0.0000003590	0.0000582119	0.0002857252

Table 5.2.1.2 – *Performance comparison adding columns to dataframe in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	0.3118944%	100.0000000%	453.0950019%
1,000	0.3963421%	100.0000000%	522.0883609%
10,000	0.2580262%	100.0000000%	474.1329312%
100,000	0.2948321%	100.0000000%	482.6797905%
1,000,000	0.6167815%	100.0000000%	490.8367127%

Table 5.2.1.3 – *Analysis of variance of execution time for add column task*

Sample Size	F Statistic	P-value
100	319.6036982	2.9912E-54
1000	703.6291298	5.2586E-76
10000	1393.337743	2.77356E-96
100000	886.1248127	9.71989E-83
1000000	949.0999733	9.09379E-85

5.2.2 Adding Rows to Dataframe:

Table 5.2.2.1 – *Mean execution time for adding rows to dataframe in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0000002065	0.0006020982	0.0003262713
1,000	0.0000002039	0.0009595808	0.0005502062
10,000	0.0000001957	0.0065670171	0.0046304645
100,000	0.0000001948	0.0504913999	0.0284873130
1,000,000	0.0000002120	0.6300234609	0.4264532398

Table 5.2.2.2 – *Performance comparison adding rows to dataframe in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	0.0343034%	100.0000000%	54.1890512%
1,000	0.0212447%	100.0000000%	57.3381832%
10,000	0.0029807%	100.0000000%	70.5109260%
100,000	0.0003858%	100.0000000%	56.4201291%
1,000,000	0.0000336%	100.0000000%	67.6884698%

Table 5.2.2.3 – *Analysis of variance of execution time for add rows task*

Sample Size	F Statistic	P-value
100	219.589801	7.03992E-45
1000	1728.53201	7.4744E-103
10000	332.7382215	2.67136E-55
100000	196.7492013	2.73801E-42
1000000	705.3809479	4.45651E-76

5.2.3 Left join:

Table 5.2.3.1 – *Mean execution time for left join to dataframe in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0000003047	0.0019212360	0.0048947043
1,000	0.0000003899	0.0041299382	0.0058566790
10,000	0.0000003161	0.0386587562	0.0070598526
100,000	0.0000003556	0.5244205575	0.0288984245
1,000,000	0.0000002032	5.7803973284	0.2307397870

Table 5.2.3.2 – *Performance comparison left join to dataframe in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	0.0158596%	100.0000000%	254.7685116%

1,000	0.0094413%	100.0000000%	141.8103298%
10,000	0.0008176%	100.0000000%	18.2619756%
100,000	0.0000678%	100.0000000%	5.5105438%
1,000,000	0.0000035%	100.0000000%	3.9917634%

Table 5.2.3.3 – *Analysis of variance of execution time for left join task*

Sample Size	F Statistic	P-value
100	403.5866659	1.97342E-60
1000	504.0151501	1.57475E-66
10000	1346.427798	3.02377E-95
100000	4622.224936	2.0044E-133
1000000	5050.487086	3.2804E-136

5.2.4 Right join:

Table 5.2.4.1 – *Mean execution time for right join to dataframe in seconds*

Sample Size	data.table	Native R	Tidyverse
100	0.0000002715	0.0015048951	0.0047442329
1,000	0.0000002172	0.0032576938	0.0036654969
10,000	0.0000003534	0.0348239021	0.0086731605
100,000	0.0000002602	0.3973108405	0.0234124074
1,000,000	0.0000004012	5.3114132951	0.2605791963

Table 5.2.4.2 – *Performance comparison right join to dataframe in seconds (considering Native R as standard)*

Sample Size	data.table	Native R	Tidyverse
100	0.0180398%	100.0000000%	315.2533897%
1,000	0.0066685%	100.0000000%	112.5181538%
10,000	0.0010148%	100.0000000%	24.9057687%
100,000	0.0000655%	100.0000000%	5.8927180%
1,000,000	0.0000076%	100.0000000%	4.9060237%

Table 5.2.4.3 – *Analysis of variance of execution time for right join task*

Sample Size	F Statistic	P-value
100	1409.401	1.25E-96
1000	2822.611	5.4E-118
10000	1400.476	1.94E-96
100000	2834.344	4E-118
1000000	10295.62	1E-158

6. Conclusion & Discussion:

Analysis of performance data for the task of sorting shows for the lower sample sizes (<104 rows) performance of Tidyverse was not as per mark compared with Native R and data.table. data.table performed better than native R in all the sorting task. While in higher sample sizes (>105 rows) Tidyverse has performed better than Native R but lacks in performance in comparison with data.table.

Analysis of variance shows a significant (p-value < 0.05) difference in mean performance time of frameworks in different sorting task (specified under figure 3.2) considered in study. Post-hoc study (HSD-Tukey Test) of pairwise performance comparison shows the same findings and there were significant (p-value < 0.05) differences between the performance of pairs of frameworks.

While analyzing the joining task data, adding columns to a dataset task data shows that performance of Native R is better than Tidyverse for all the dataset sizes in the scope but data.table performance found significantly better compared to native R. Analysis of variance shows there is a significant difference between the performance of all the platforms and pairwise comparison also shows the same.

Adding rows to existing dataset task shows performance of Native R was lagging behind Tidyverse and performance of data.table was dominating other two frameworks. The performance difference was proven significant (p-value < 0.05) in analysis of variance and pair-wise comparisons.

Joining tasks (Left & Right Join) performance data shows same patters like sorting task and for lower samples Tidyverse was performing inefficient then Native R but data.table has outperformed both the frameworks. The same established using analysis of variance and pairwise comparisons.

Conflict of Interest: Authors do not have any conflict of interest as there is no external/internal funding used to complete this work.

References:

[1] R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[2] Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." _Journal of Open Source Software_, *4*(43), 1686. doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.

[3] Dowle M, Srinivasan A (2021). _data.table: Extension of `data.frame`_. R package version 1.14.2, <https://CRAN.R-project.org/package=data.table>.

[4] R Core Team. (2021). object.size: Estimate the Size of R Objects (R version 4.1.0). R Foundation for Statistical Computing. https://www.rdocumentation.org/packages/base/versions/4.1.0/topics/object.size

[5] Wickham, H., & Csárdi, G. (2020). nycflights13: Flights that Departed NYC in 2013. R package version 1.1.0. https://CRAN.R-project.org/package=nycflights13

[6] Müller, K., Wickham, H., & François, R. (2021). tibble: Simple Data Frames (R version 4.1.0). RStudio. https://tibble.tidyverse.org

[7] Montgomery, D. C. (2017). Design and Analysis of Experiments. John Wiley & Sons.

[8] Agresti, A., & Franklin, C. (2018). Statistics: The Art and Science of Learning from Data

[9] Field, A., Miles, J., & Field, Z. (2012). Discovering Statistics Using R. SAGE Publications

[10] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media.

[11] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2012). Introduction to Linear Regression Analysis. John Wiley & Sons.

[12] Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2004). Applied Linear Statistical Models. McGraw-Hill.

[13] Draper, N. R., & Smith, H. (1998). Applied Regression Analysis (3rd ed.). Wiley-Interscience.

[14] Wickham, H. (2021). Memory. Advanced R. http://adv-r.had.co.nz/memory.html