Data Frame Operations in R Using Pipe Command with *tidyverse* and *dplyr* packages vs Base R Functions

Jomari Manto   
*College of Computer Studies & Engineering*  
*Jose Rizal University*Mandaluyong City, Philippines  
[jomari.manto@my.jru.edu](mailto:jomari.manto@my.jru.edu)

*Abstract*—This report outlines various data frame operations in R using the pipe (%>%) operator, primarily with the *****tidyverse***** and *****dplyr***** packages. The report covers fundamental data manipulation tasks such as creating and loading data frames, as well as adding or deleting variables and other components. These operations will be demonstrated using simulated datasets, showcasing the practical application of the *****tidyverse***** and *****dplyr***** packages. A discussion of how these functions work compared to Base R functions is also included in the Results, and Discussion sections.(*Abstract*)

Keywords—DataFrame operations, pipe command in r, tidyverse package, dplyr package, Base R functions (keywords)

# Introduction: Data Frames

Data frames are one of the fundamental structures in R used in storing and manipulating data for the purpose of data analysis. It comes in various data types such as numbers(integers), text(strings), and other data types(Boolean, doubles).

Similar to an Excel spreadsheet, data frames in R also use a collection of columns and rows to host data. In the context of data analysis, it is imperative for analysts to master creating, and performing different operations on R data frames such as creating a data frame from scratch directly in R, or creating a data frame using an a file sourced externally such as a comma-separated values(csv) files, Excel files, et cetera.

Manipulating the values inside these data frames is of equal importance, in order for analysts to reach the ultimate goal of gathering useful insights from their data.

# Methodology

## Creating DataFrames Using the data.frame() Command

There’s more than one way to create data frames in R. The most common method is to use the command *data.frame()*which uses vectors (list of variables) as input, combining the values to create a data frame. Figure 1 shows how the *data.frame()* command used to create a data frame in R.

A screenshot of a computer code

AI-generated content may be incorrect.

Fig. 1. A data frame created using the *data.frame()* command in R.

Certainly, while there are advantages of using the command in starting with data frames in R, there can be certain drawbacks, particularly when working with larger data sets. In those cases, the *tidyverse*, and the *dplyr* packages are effective tools that can be used to process data frames in R.

## Creating & Performing DataFrame Operations Using the tidyverse, and dplyr packages

To ease the process of performing operations in data frames, R packages developed by various people are available for installation and usage. Both the *tidyverse* and *dplyr* are core R packages that analysts use to process data frames.

The *tidyverse,* a collection of related R packages contains functions designed to perform tasks related to data science. Within the package is the *dplyr* package specifically designed to perform data manipulation tasks. Listed in the table below are key functions of the *dplyr* package. Please note, that the below list is non-exhaustive.

Table 1. Key Functions: dplyr Package

| Command | Key Function |
| --- | --- |
| filter() | Subset, i.e. filter observations to meet specific conditions |
| select() | Choose columns that you want to display. |
| mutate() | Modify existing variables in the data frame and is also used to add new columns while keeping the original data. |
| transmute() | The *transmute()* command is used to add new variables and/or modify existing, however its output will only include the new variables, and will drop the original data. |
| arrange() | Arrange variable to the desired order |

Alongside with the pipe command (%>%), multiple data frame operations were executed starting with creating data frame to manipulating and modifying variables within the data frame. To install, simply use the command the *install.packages("tidyverse")*.

Table 2 below shows the data frame operations executed and the corresponding commands that were used.

Table 2. Data Frame Operations & Commands Used

| Operation | Command |
| --- | --- |
| 1. Creating a data frameafl.finalists | tibble() |
| 2. Loading a data frame (.csv file) | read\_csv() |
| 3. Displaying structure and summary statistics | view(), head(), str(),print(), glimpse(), summary() |
| 4. Accessing rows and columns | select(), filter(), slice() |
| 5. Selecting subsets of data | select(), filter(), |
| 6. Editing data frames | mutate(), rename() |
| 7. Adding extra rows and columns | mutate() |
| 8. Adding new variables to data frame | mutate() |
| 9. Describing the date frame | summary(), str(), head() |

## C. Pipe Operator and Assign Operator

## The pipe operator (%>%) and assign operator (<-) are two of the commonly used operators in R. The assign operator is used to assign values in R, while the pipe operator enables to “chain multiple operations. It’s primary benefit is to address the complexity of nested functions, which can be quite cumbersome, and if you are processing large quantities of data, the last thing you want is to used long, complex commands.

Some users may find the terminology complex, especially beginners, so here’s a simple analogy of how the pipe command makes data frame operations in R simpler –imagine you’re making a table on an assembly line:

Worker 1: Cuts the wood.

Worker 2: Paints the wood.

Worker 3: Puts the wood together to make a table.

Worker 4: Adds the finishing touches.

Each worker does just one job, and then hands the piece along to the next worker. You don’t need to carry it around, they just pass it on.

How It Relates to the Pipe Operator:

The data is like the piece of wood.

Each worker is like a function (e.g., mutate(), filter(), select()).

The pipe operator %>% is like handing the piece of wood from one worker to the next.

Instead of doing everything in one big messy step, each function takes care of one thing and passes the result to the next. The pipe operator helps pass the data through each function one by one, just like the workers on the assembly line.

# Results

Performing data frame operations using the functions within the *tidyverse* package compared to Base R functions is significantly more efficient.

For context, the data frame used, *online\_retail* has a total of 20 observations, and 10 variables. See image in figure 2.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 2. Data frame *online\_retail.*

From the data frame, in order to subset the *online\_retail* dataset and extract the rows where Country is "South Africa" and the Quantity is greater than 5, using the Base R functions, the command to be used will be:

*subset\_base <- online\_retail[online\_retail$Country == "South Africa" & online\_retail$Quantity > 5, ]*

Explanation:

* *online\_retail$Country == "South Africa"* checks if the Country column is "South Africa".
* *online\_retail$Quantity > 5* checks if the Quantity is greater than 5.
* The *&* operator is used to combine both conditions.
* The result is passed into the square brackets *[ ]* to return the subset of rows that meet both conditions.

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 3. Using Base R functions to subset a data frame.

Using the *tidyverse* package, to perform the same operation on the data frame, we only need to write the below command:

*subset\_tidyverse <- online\_retail %>% filter(Country == "South Africa", Quantity > 5) print(subset\_tidyverse)*

* The *filter()* function is used to filter the rows where Country is *"South Africa"* and *Quantity* is greater than *5*.
* The *%>%* (pipe operator) is used to pass the *online\_retail* dataset to the *filter()* function.
* The conditions inside *filter()* are separated by commas, which are implicitly combined with an *AND* logic.

*A screenshot of a computer

AI-generated content may be incorrect.*

Fig. 4. Using *tidyverse* functions to subset a data frame.

With the Base R, logical conditions inside square brackets are used. In *tidyverse (dplyr),* the *filter()* function is used, which is more readable and user-friendly, especially when dealing with complex conditions. Both approaches will give you the same subset of rows where Country is "South Africa" and Quantity is greater than 5.

For another example, supposed we are to calculate the mean, median, and the standard deviation of the Quantity column from our data frame. In Base R, the built-in functions will be used:

* *mean\_quantity <- mean(online\_retail$Quantity)*
* *median\_quantity <- median(online\_retail$Quantity)*
* *sd\_quantity <- sd(online\_retail$Quantity)*

A screenshot of a computer

AI-generated content may be incorrect.

Fig. 5. Statistical operations using Base R functions.

Using the *tidyverse* package, with *dplyr,* the same operations can be performed by the *summarise()* function:

*summary\_stats <- online\_retail %>%*

*summarise(*

*mean\_quantity = mean(Quantity, na.rm=TRUE),*

*median\_quantity = median(Quantity, na.rm=TRUE),*

*sd\_quantity = sd(Quantity, na.rm = TRUE) )*

*print(summary\_stats)*

##### *A screenshot of a computer AI-generated content may be incorrect.*

Fig. 6. Statistical operations using *tidyverse(dplyr)* package

With Base R functions, each statistical value is calculated individually, whereas in the tidyverse(dplyr) package, multiple statistics can be calculated using one function – the *summarise()* function which provides a more compact, organized, and readable approach, especially if you need to calculate several measurements at once.

##### Discussion

Both methods of performing data frame operations have their own advantages. Depending on the operations to be performed, each user may freely choose between the two methods to fit the need. It is also important to note that both methods also have limitations. For instance, t*idyverse* functions rely on external packages, and certain situations where installation process is restricted could limit using these functions.

Base R on the other hand has no external dependencies and provides more freedom when performing tailor–fit tasks, and more complex data frame manipulations that might not be available in the package.

##### Conclusion

Learning to utilize both Base R functions and packages is more beneficial, and is more holistic compared to completely sticking with one method. Albeit overwhelming, gradually learning function on a case-to-case basis would be a good start.

While the packages such as the *tidyverse* provides benefits such as convenience, consistency and uniformity, user-friendly, as well as additional functionalities. Mastering Base R functions also provides a huge advantage. Afterall, R packages are typically created using the Base R functions, and R programming concepts. Together, both methods can complement and/or supplement each other.

##### References

1. G. GeeksforGeeks, "DataFrame Operations in R," *GeeksforGeeks*. [Online]. Available: <https://www.geeksforgeeks.org/dataframe-operations-in-r/>. [Accessed: 13-Feb-2025]. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
2. DataCamp, "What is Data Wrangling?," DataCamp. [Online]. Available: <https://www.datacamp.com/blog/what-is-data-wrangling?utm_source=google&utm_medium=paid_search&utm_campaignid=19589720824&utm_adgroupid=152984013534&utm_device=c&utm_keyword=&utm_matchtype=&utm_network=g&utm_adpostion=&utm_creative=726015683646&utm_targetid=dsa-2222697811358&utm_loc_interest_ms=&utm_loc_physical_ms=1011171&utm_content=DSA~blog~Data-Science&utm_campaign=230119_1-sea~dsa~tofu_2-b2c_3-row-p2_4-prc_5-na_6-na_7-le_8-pdsh-go_9-nb-e_10-na_11-na-jan25&gad_source=1&gclid=CjwKCAiAzPy8BhBoEiwAbnM9O9_QYeTA_56H_oV8Cq3ocwH9uyF4PjWIdi7xp8BDrd4Hm9G17uHeWRoCtN8QAvD_BwE>. [Accessed: 13-Feb-2025].
3. J. Scott, "Importing Data," Bookdown. [Online]. Available: <https://bookdown.org/jgscott/DSGI/lesson_data.html#importing_data>. [Accessed: 13-Feb-2025].
4. DataCamp, "Pipe R Tutorial," DataCamp. [Online]. Available: <https://www.datacamp.com/tutorial/pipe-r-tutorial>. [Accessed: 13-Feb-2025].
5. R Studio, "Tibbles," R for Data Science. [Online]. Available: <https://r4ds.had.co.nz/tibbles.html>. [Accessed: 13-Feb-2025].
6. J. Scott, "Data Frames: Cases and Variables," Bookdown. [Online]. Available: https://bookdown.org/jgscott/DSGI/lesson\_data.html#data-frames-cases-and-variables. [Accessed: 13-Feb-2025].
7. W3Schools, "R Data Frames," W3Schools. [Online]. Available: <https://www.w3schools.com/r/r_data_frames.asp>. [Accessed: 13-Feb-2025].
8. G. GeeksforGeeks, "How to Create DataFrame in R," GeeksforGeeks. [Online]. Available: <https://www.geeksforgeeks.org/how-to-create-dataframe-in-r/>. [Accessed: 13-Feb-2025].
9. Tidyverse, "Tidyverse Packages," Tidyverse. [Online]. Available: <https://www.tidyverse.org/packages/>. [Accessed: 13-Feb-2025].
10. RStudio, "dplyr," dplyr - Tidyverse. [Online]. Available: <https://dplyr.tidyverse.org/>. [Accessed: 13-Feb-2025].
11. M. Shieh, "Base R vs. Tidyverse," Marcy Shieh's Blog. [Online]. Available: <https://marcyshieh.com/ps811-materials/base-r-vs-tidyverse.html>. [Accessed: 13-Feb-2025].