What are the three key differences between data.table and data frame in R?

Authored by

stats writer

June 26, 2024

Data.table and data frame are two commonly used data structures in the R programming language. While they share some similarities, there are three key differences that set them apart.

1. Speed and efficiency: Data.table is known for its superior speed and efficiency compared to data frame. This is because data.table is optimized for handling large datasets, making it more suitable for handling big data. On the other hand, data frame is more suitable for smaller datasets.

2. Syntax and functionality: The syntax and functionality of data.table and data frame are also different. Data.table uses the concept of keys and joins to manipulate data, which allows for faster and more efficient data manipulation. Data frame, on the other hand, uses the traditional R syntax for data manipulation.

3. Memory management: Another key difference between data.table and data frame is how they handle memory. Data.table is designed to minimize memory usage, making it more efficient for handling large datasets. Data frame, on the other hand, can consume a significant amount of memory, especially when dealing with large datasets.

In summary, data.table and data frame differ in terms of speed, syntax, and memory management. Choosing between the two will depend on the size and complexity of the dataset and the specific requirements of the analysis or project.

data.table vs. data frame in R: Three Key Differences

In the R programming language, a data.frame is part of base R.

Any data.frame can be converted to a data.table by using the setDF function from the data.table package.

A data.table offers the following benefits over a data.frame in R:

1. You can use the function from the data.table package to read a file into a data.table *much* faster than base R

functions such as , which read files into a data.frame.

2. You can perform operations (such as grouping and aggregating) on a data.table *much* faster than a data.frame.

3. When printing a data.frame to a console, R will attempt to display every single row from the data.frame. However, a data.table will only display the first 100 rows, which can prevent your session from freezing or crashing if you're working with a massive dataset.

The following examples illustrate these differences between data.frames and data.tables in practice.

Difference #1: Faster Importing with fread Function

The following code shows how to import some data frame with 10,000 rows and 100 columns using the fread function from the data.table package and the read.csv function from base R:

```
library(microbenchmark)
library(data.table)

#make this example reproducible
```

```
set.seed(1)

#create data frame with 10,000 rows and 100 columns
df <- as.data.frame(matrix(runif(10^4 * 100), nrow = 10^4))

#export CSV to current working directory
write.csv(df, "test.csv", quote = FALSE)

#import CSV file using fread and read.csv and time how long it takes
results <- microbenchmark(
read.csv = read.csv("test.csv", header = TRUE, stringsAsFactors = FALSE),
fread = fread("test.csv", sep = ",", stringsAsFactors = FALSE),
times = 10)

#view results
results

Unit: milliseconds
expr min lq mean median uq max neval cld
read.csv 817.1867 892.8748 1026.7071 899.5755 926.9120 1964.0540 10 b
```

fread 113.5889 116.2735 136.4079 124.3816 136.0534 211.7484 10 a

From the results we can see that fread is roughly 10 times faster at importing this CSV file compared to the read.csv function.

Note that this difference will be even greater for larger datasets.

Difference #2: Faster Data Manipulation with data.table

In general, data.table can also perform any data manipulation task much faster than a data.frame.

For example, the following code shows how to calculate the mean of one variable, grouped by another variable in both a data.table and data.frame:

```
library(microbenchmark)
library(data.table)

#make this example reproducible
set.seed(1)

#create data frame with 10,000 rows and 100 columns
```

```
d_frame <- data.frame(team=rep(c('A', 'B'), each=5000),
points=c(rnorm(10000, mean=20, sd=3)))

#create data.table from data.frame
d_table <- setDT(d_frame)

#calculate mean of points grouped by team in
data.frame and data.table
results <- microbenchmark(
mean_d_frame = aggregate(d_frame$points,
list(d_frame$team), FUN=mean),
mean_d_table = d_table,
times = 10)

#view results
results

Unit: milliseconds
expr min lq mean median uq max neval cld
mean_d_frame 2.9045 3.0077 3.11683 3.1074 3.1654
3.4824 10 b
mean_d_table 1.0539 1.1140 1.52002 1.2075 1.2786
3.6084 10 a
```

From the results we can see that data.table is about

three times faster than data.frame.

Difference #3: Fewer Printed Lines with data.table

When printing a data.frame to a console, R will attempt to display every single row from the data.frame.

However, a data.table will only display the first 100 rows, which can prevent your session from freezing or crashing if you're working with a massive dataset.

For example, in the following code we create both a data frame and a data.table with 200 rows.

When printing the data.frame, R will attempt to print every single row while printing the data.table will only show the first five rows and last five rows:

```
library(data.table)

#make this example reproducible
set.seed(1)

#create data frame
d_frame <- data.frame(x=rnorm(200),
y=rnorm(200),
z=rnorm(200))
```

```
#view data frame
d_frame

x y z
1 -0.055303118 1.54858564 -2.065337e-02
2 0.354143920 0.36706204 -3.743962e-01
3 -0.999823809 -1.57842544 4.392027e-01
4 2.586214840 0.17383147 -2.081125e+00
5 -1.917692199 -2.11487401 4.073522e-01
6 0.039614766 2.21644236 1.869164e+00
7 -1.942259548 0.81566443 4.740712e-01
8 -0.424913746 1.01081030 4.996065e-01
9 -1.753210825 -0.98893038 -6.290307e-01
10 0.232382655 -1.25229873 -1.324883e+00
11 0.027278832 0.44209325 -3.221920e-01
...
#create data table
d_table <- setDT(d_frame)

#view data table
d_table

x y z
1: -0.05530312 1.54858564 -0.02065337
2: 0.35414392 0.36706204 -0.37439617
```

3: -0.99982381 -1.57842544 0.43920275
4: 2.58621484 0.17383147 -2.08112491
5: -1.91769220 -2.11487401 0.40735218

196: -0.06196178 1.08164065 0.58609090
197: 0.34160667 -0.01886703 1.61296255
198: -0.38361957 -0.03890329 0.71377217
199: -0.80719743 -0.89674205 -0.49615702
200: -0.26502679 -0.15887435 -1.73781026

This is a benefit that data.table offers compared to data.frame, especially when working with massive datasets that you don't want to accidently print to the console.