How can I sort the rows of a DataFrame by a specific column value in R?

Authored by

stats writer

June 24, 2024

To sort the rows of a DataFrame by a specific column value in R, you can use the "order" function. This function takes in the column you want to sort by as well as the DataFrame and returns a vector which specifies the order of the rows. This vector can then be used to rearrange the rows of the DataFrame using the "" indexing operator. This process can also be simplified using the "arrange" function from the "dplyr" package, which allows you to sort by a specific column in ascending or descending order. Both methods will result in a DataFrame with the rows sorted according to the values in the specified column.

How to sort data frame in R? To sort data frame by column values use the order() function. By default, the sorting function performs in ASCENDING order and provides an option to sort in descending order. Also, by default, all NA values on the sorting column are kept at the last and you can change this behavior by using optional params.

Key Points -

In this article, I will explain how to sort dataframe by using the above key points along with using the following methods.

1. Quick Examples of Sorting DataFrame

Following are quick examples of how to sort DataFrame by column value in ascending and descending order in R programming.

```
# Below are the quick examples of sorting data frame

# Example 1: Sort data frame by specific column
df2 <- emp_df

# Example 2: Sort by multiple columns
df2 <- df

# Example 3: Sort descending order
df2 <- df

# Example 4: Sort by putting NA top
df2 <- df

# Example 5: Load dplyr library
library(dplyr)
df2 <- df %>% arrange(price)
df2 <- df %>% arrange(desc(price), desc(name) ))
```

```
# Example 6: Load data.table library
library("data.table")
df2 <- setorder(df,price)
```

Let's create an R DataFrame with the columns of `"id"`, `"name"`, `"price"`, and `"publish_date"`. Apply different sorting methods to the data frame and get the new data frame by reordering the rows based on values of single/multiple columns.

```
# Create Data Frame
df = data.frame(id=c(11,22,33,44,55),
name=c("spark","python","R","jsp","java"),
price=c(144,NA,321,567,567),
publish_date= as.Date(
c("2007-06-22", "2004-02-13", "2006-05-18",
"2010-09-02","2007-07-20"))
)
df
```

Yields below output.

```
  id   name price publish_date
1 11  spark   144   2007-06-22
2 22 python    NA   2004-02-13
3 33      R   321   2006-05-18
4 44    jsp   567   2010-09-02
5 55   java   567   2007-07-20
```

2. Sort DataFrame in R using order() Function

The `order()` is a base function that is used to sort the data frame in R based on column value, this function can also be used to sort vectors. It takes the ordered column indices, so we have to use - index, and inside this, we can apply the `order()` function. Hence this will return the column.

2.1 Syntax of order() Function

Following is the syntax of the order() function.

```
# order() syntax
order(data, na.last = TRUE, decreasing = FALSE)
```

2.2 Example of Sort DataFrame by Column Value

Let's apply the order() function to sort the R dataframe by column value in ascending order. The following example sorts the data by column `price`. Since this function takes the vector as an argument, use `df$price` it as an argument. Note that in R, every column in DataFrame is a vector.

```
# Sort DataFrame by specific column value
df2 <- df
df2
```

Yields below output.

	id	name	price	publish_date
1	11	spark	144	2007-06-22
3	33	R	321	2006-05-18
4	44	jsp	567	2010-09-02
5	55	java	567	2007-07-20
2	22	python	NA	2004-02-13

4. By Multiple Columns

If you want to sort by multiple columns, you can pass all columns/variables into the order() function. It will return the new data frame by re-ordering the rows of the original data frame `df` based on specified columns.

In this case, Pass `price` and `name` columns into the order() function. It sorts first by `price` and then by `name`.

The result is a new data frame `df2` with rows arranged in ascending order based on the `price` column. In case of ties in the `price`, the rows are further sorted based on the `name` column.

```
# Sort by multiple columns
df2 <- df
df2
```

Output:
id name price publish_date
1 11 spark 144 2007-06-22
3 33 R 321 2006-05-18
5 55 java 567 2007-07-20
4 44 jsp 567 2010-09-02
2 22 python NA 2004-02-13

6. Sort DataFrame by Descending order

By default, sorting happens in ascending or increasing order, for string columns, the sort happens in alphabetical order (A to Z). You can sort the data.frame in descending order by using `decreasing=TRUE`.

```
# Sort descending order
df2 <- df
df2
```

Output:
id name price publish_date
1 11 spark 144 2007-06-22
3 33 R 321 2006-05-18
2 22 python NA 2004-02-13
4 44 jsp 567 2010-09-02
5 55 java 567 2007-07-20

When arranging the rows you can also prefix with a minus sign to perform sorting in decreasing order. This allows you to sort one column in ascending and another column in descending order.

7. Sort by having NA First

By default rows with NA on a sorting column will be put at the end in R, however, you can change this behavior and put rows with NA on top of the dataframe by using `na.last=FALSE`.

```
# Sort by putting NA at first
```

```
df2 <- df
df2
```

Output:
id name price publish_date
2 22 python NA 2004-02-13
4 44 jsp 567 2010-09-02
5 55 java 567 2007-07-20
3 33 R 321 2006-05-18
1 11 spark 144 2007-06-22

8. Sort DataFrame using dplyr Package

arrange() function from dplyr package is also used to arrange the values in an ascending or descending order. To use the arrange() function, you have to install dplyr first using install.packages('dplyr') and load it using `library(dplyr)`.

All functions in dplyr package take `data.frame` as a first argument. When we use `dplyr` package, we mostly use the infix operator `%>%` from `magrittr`, it passes the left-hand side of the operator to the first argument of the right-hand side of the operator. For example, `x %>% f(y)` converted into `f(x, y)` so the result from the left-hand side is then "piped" into the right-hand side.

```
# Load dplyr library
library(dplyr)
df2 <- df %>% arrange(price)
df2
```

To sort columns in descending order

```
# Sort descending order
df2 <- df %>% arrange(desc(price))
df2
```

9. Sort DataFrame using data.table Package

You can also use the setorder() function from the data.table to perform sorting on DataFrame columns. This function takes the data.frame object and column as input and return a new DataFrame after sorting by the specified column.

```
# Load data.table library
library("data.table")
df2 <- setorder(df,price)
df2
```

Conclusion

In this article, I have explained how to use the order() function to sort the DataFrame values by single/multiple columns in R and how to sort by descending order, keeping all NA values first. Finally converted using the arrange() function from the dplyr package and the setorder() function from the data.table to order data.frame by specified columns.

Similarly, you can also sort by date column, when performing an order on a date make sure the column is in date type.

Related Articles

References