What is the Pipe operator in R and how is it used? Can you provide some examples?

Authored by

stats writer

June 23, 2024

The Pipe operator in R, also known as the `%>%` symbol, is a useful tool for manipulating and transforming data. It allows for a more efficient and readable coding style by allowing functions to be chained together in a series of steps.

The Pipe operator is used to pass the output of one function as the input of another function, eliminating the need for intermediate variables. This enables a more streamlined and concise code structure.

For example, instead of writing the code as follows:
```
output % function1() %>% function2() %>% function3()
```

This allows for a clearer and more logical flow of data transformation steps. The Pipe operator can also be used with complex functions that take multiple arguments, making it easier to read and understand the code.

Overall, the Pipe operator in R is a valuable tool for data manipulation and can greatly improve the efficiency and readability of code.

Pipe %>% in R is the most used operator that was introduced in **magrittr** package by Stefan Milton Bache. The pipe operator %>% is used to express a sequence of multiple operations, for example, the output of one function or expression is passed to another function as an argument.

Key Points of using Pipe in R

Pipe %>% in R is introduced in `magrittr` packageWhen using the `tidyverse` package, this %>% pipe operator is automatically available for you to use.It takes the output of one function and passes it into another function as an argumentIt works with a function that takes one input.If a function needs two inputs then it can't be used.

1. What is Pipe Operator in R - Introduction

Pipe in R is an infix operator that was introduced in **magrittr** package by Stefan Milton Bache, which is used to pass the output of one function as an input to another function which ideally makes our code easily readable and efficient. In other words pipe operator %>% is used to express a sequence of multiple operations in an elegant way.

2. Does Pipe Exists in other Languages

If you are familiar with Linux, you would probably know the pipe operator | that is used to pass the output of one command to another. So the Pipe is nothing new to the programming its existence

has been there for a while.

```
# Pipe | operator in Linux
ls -lrt | grep new_file | <additional comamnds>
```

3. Why do we need Pipe in R

When you have complex code to write in R Programming, sometimes you nest the operations which result in unreadable R code and it's hard for others to understand. By using the pipe %>% operator to can avoid writing complex code to some extent by chaining them.

Also, By using the pipe %>% operator you can save the memory footprint of your R program. hence, it's been used in many R packages now. You might ask how it saves memory.

For example, imagine you have 3 function calls and the result of each function is passed as input to another function, when you don't have %>% operator you would store the result of each function into a variable and use this variable on another function, so here we are keeping 3 variable in memory with the data. By using a pipe, you can avoid this intermediate variable and chain the functions using pipe operator.

3.1 Example without Pipe in R

Following is a basic example of using 3 functions, and the output of each function is passed as input to another function.

```
# Simile example with out Pipe operator
# add function
add <- function(x,y) {
return (x + y)
}
# multiply function
mul <- function(x,y) {
return (x * y)
}
# div function
div <- function(x,y) {
return (x / y)
}
```

calling functions sequentially

```
res1 <- add(2,4)
res2 <- mul(res1,8)
res3 <- div(res2,2)
print(res3)

# Output
# 24
```

Note that you can also write this by nesting the functions, since our examples are small it is still okay but imagine you have functions that take several arguments, and nesting these becomes your R code unreadable.

```
# Using nesting functions
res <- div(mul(add(2,4),8),2)
print(res)

# Output
24
```

4. How to use Pipe Operator in R

When the Pipe operator %>% is used in an R expression or function, 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 left-hand side is then "piped" into the right-hand side. This pipe can be used to write multiple operations that you can read left-to-right.

Let's see with an example.

```
# Using with pipe
res <- add(2,4) %>% mul(8) %>% div(2)
print(res)

# Output
# 24
```

4. Using Pipe with Dplyr Package

The dplyr is a package that provides a grammar of data manipulation and provides the most used

verbs that help data science analysts to solve the most common data manipulation. By using methods from this package over the R base function results in better performance.

In order to use dplyr verbs, you have to install the package first using `install.packages('dplyr')` and load it using `library(dplyr)`.

All verbs in `dplyr` package take data.frame as a first argument. When we use `dplyr` package, we mostly use the infix operator `%>%`. Let's see with an example.

```
# Create DataFrame
df <- data.frame(
id = c(10,11,12,13),
name = c('sai','ram','deepika','sahithi'),
gender = c('M','M','F','F'),
dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16')),
state = c('CA','NY',NA,NA),
row.names=c('r1','r2','r3','r4')
)
df
```

```
# Load dplyr library
library('dplyr')

# filter() by row name & select id and name columns
df2 <- df %>% filter(rownames(df) == 'r3') %>% select(c('id','name'))
print(df2)
```

Yields below output

```
> # Create DataFrame
> df <- data.frame(
+   id = c(10,11,12,13),
+   name = c('sai','ram','deepika','sahithi'),
+   gender = c('M','M','F','F'),
+   dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16')),
+   state = c('CA','NY',NA,NA),
+   row.names=c('r1','r2','r3','r4')
+ )
> df
   id    name gender        dob state
r1 10     sai      M 1990-10-02    CA
r2 11     ram      M 1981-03-24    NY
r3 12 deepika      F 1987-06-14  <NA>
r4 13 sahithi      F 1985-08-16  <NA>
> # filter() by row name
> df2 <- df %>% filter(rownames(df) == 'r3') %>% select(c('id','name'))
> print(df2)
   id    name
r3 12 deepika
```

5. Limitations of using Pipe

The following are limitations of the Pipe in R.

It takes the output of one function and passes it into another function as an argumentIt works with a function that takes one input.If a function needs two inputs then it can't be used.

6. Conclusion

In this article you have learned what is pipe operator in R, and how and when to use it. To express a sequence of multiple operations you can use pipe %>%. This takes the output of one function or expression and passes it to another function as an argument.

The complete example explained above is available at GitHub R Examples.

Related Articles

Sort Table in R with ExamplesR Group by Count With ExamplesR Group by Sum With ExamplesR Group by Mean With ExamplesDates and Times in R with ExamplesFor Loop in R - With ExamplesR solve() Equation with ExamplesSparklyr Sort DataFrame with ExamplesFilter in

sparklyr | R Interface to SparkSparklyr join explained with examplesExplained apply Functions in R with ExamplesR Count Frequency of All Unique Values in Vector