How can the pipe operator be used in R, and what are some examples?

Authored by

stats writer

June 27, 2024

The pipe operator, denoted by "%>%" in R, is a useful tool for streamlining code and improving readability. It allows for the sequential execution of multiple functions, with the output of one function serving as the input of the next. This eliminates the need for nested functions and improves the flow of code.

For example, instead of writing "mean(sqrt(abs(x)))", the pipe operator allows for "x %>% abs() %>% sqrt() %>% mean()". This makes the code easier to understand, especially when dealing with multiple functions and complex operations.

Another example is in data manipulation, where the pipe operator can be used to efficiently perform multiple operations on a dataset. For instance, "data %>% filter(age > 18) %>% select(name, occupation) %>% arrange(name)" would filter the data for individuals over the age of 18, select their names and occupations, and then arrange them alphabetically by name.

In summary, the pipe operator in R is a powerful tool for improving the efficiency and readability of code, especially when dealing with multiple functions and complex operations. Its use can greatly enhance the coding experience and make data manipulation and analysis more streamlined.

Use the Pipe Operator in R (With Examples)

You can use the pipe operator (%>%) in R to "pipe" together a sequence of operations.

This operator is most commonly used with the package in R to perform a sequence of operations on a data frame.

The basic syntax for the pipe operator is:

df %>%
do_this_operation %>%
then_do_this_operation %>%

then_do_this_operation ...

The pipe operator simply feeds the results of one operation into the next operation below it.

The advantage of using the pipe operator is that it makes code extremely easy to read.

The following examples show how to use the pipe operator in different scenarios with the built-in dataset in R.

#view first six rows of mtcars dataset
head(mtcars)

mpg cyl disp hp drat wt qsec vs am gear carb
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

Example 1: Use Pipe Operator to Summarize One Variable

The following code shows how to use the pipe (%>%) operator to group by the cyl variable and then summarize the mean value of the mpg variable:

```
library(dplyr)

#summarize mean mpg grouped by cyl
mtcars %>%
group_by(cyl) %>%
summarise(mean_mpg = mean(mpg))

# A tibble: 3 x 2
cyl mean_mpg

1 4 26.7
2 6 19.7
3 8 15.1
```

From the output we can see:

The mean mpg value for the cars with a cyl value of 4 is 26.7.The mean mpg value for the cars with a cyl value of 6 is 19.7.The mean mpg value for the cars with a cyl value of 8 is 15.1.

Notice how easy the pipe operator makes it to interpret the code as well.

Essentially, it says:

Take the mtcars data frame.Group it by the cyl variable.Then summarize the mean value of the mpg variable.

Example 2: Use Pipe Operator to Group & Summarize Multiple Variables

```
library(dplyr)

#summarize mean mpg and standard dev of hp grouped by cyl and am
mtcars %>%
group_by(cyl, am) %>%
summarise(mean_mpg = mean(mpg),
sd_hp = sd(hp))

# A tibble: 6 x 4
# Groups: cyl
cyl am mean_mpg sd_hp

1 4 0 22.9 19.7
```

2 4 1 28.1 22.7

3 6 0 19.1 9.18

4 6 1 20.6 37.5

5 8 0 15.0 33.4

6 8 1 15.4 50.2

From the output we can see:

For cars with a cyl value of 4 and am value of 0, the mean mpg value is 22.9 and the standard deviation of the hp value is 19.7.For cars with a cyl value of 4 and am value of 1, the mean mpg value is 28.1 and the standard deviation of the hp value is 22.7.

And so on.

Once again, notice how easy the pipe operator makes it to interpret the code as well.

Essentially, it says:

Take the mtcars data frame.Group it by the cyl and the am variables.Then summarize the mean value of the mpg variable and the standard deviation of the hp variable.

Example 3: Use Pipe Operator to Create New Variables

The following code shows how to use the pipe (%>%) operator along with the function from the dplyr package to create two new variables in the mtcars data frame:

```
library(dplyr)

#add two new variables in mtcars
new_mtcars <- mtcars %>%
mutate(mpg2 = mpg*2,
mpg_root = sqrt(mpg))

#view first six rows of new data frame
head(new_mtcars)
```

```
mpg cyl disp hp drat wt qsec vs am gear carb mpg2 mpg_root
1 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4 42.0 4.582576
2 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4 42.0 4.582576
3 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1 45.6 4.774935
4 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1 42.8 4.626013
5 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2 37.4 4.324350
6 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1 36.2 4.254409
```

From the output we can see:

The new mpg2 column contains the values of the mpg column multiplied by 2.The new mpg_root column contains the square root of the values in the mpg column.

Once again, notice how easy the pipe operator makes it to interpret the code as well.

Essentially, it says:

Take the mtcars data frame.Create a new column called mpg2 and a new column called mpg_root.

Related: