How and when should set.seed be used in R?

Authored by

stats writer

June 27, 2024

Set.seed is a function in the R programming language that allows for the generation of reproducible random numbers. This function is typically used when conducting statistical analyses or simulations, where the same set of random numbers is required for each run. It is recommended to use set.seed at the beginning of the code, before any random numbers are generated, to ensure consistency in the results. This is especially important when sharing code or when the same results need to be reproduced at a later time. By setting a specific seed, the same set of random numbers will be generated, providing reliable and consistent results.

How (And When) to Use set.seed in R

The set.seed() function in R is used to create reproducible results when writing code that involves creating variables that take on random values.

By using the set.seed() function, you guarantee that the same random values are produced each time you run the code.

This function uses the following basic syntax:

set.seed(seed)

where:

seed: Any number you would like.

The following examples show how to use this function in practice.

Example 1: Generate Random Values Without Using set.seed()

Suppose we use the rnorm() function to create a data frame with three variables that take on random values that follow a standard normal distribution:

```
#create data frame
df <- data.frame(var1 = rnorm(10),
var2 = rnorm(10),
var3 = rnorm(10))

#view data frame
df
```

```
var1 var2 var3
1 0.13076685 -0.32183484 0.08083558
2 0.93926332 0.92271464 1.14695121
3 1.97227368 0.01140237 0.29325751
4 1.99656555 0.26735086 1.17131155
5 -1.07893403 -0.12748185 -0.75510058
6 -0.58955485 -0.29720114 0.57928670
7 1.39367811 -1.43043111 -0.39395086
8 -0.09977302 -1.93133994 -0.66654713
9 -0.71876371 2.27999183 0.45990405
10 0.90421007 2.28077581 0.57545709
```

If we attempt to create the same data frame again using rnorm(), there is no guarantee that the values will be the same since we didn't use the set.seed() function:

```
#create data frame
df <- data.frame(var1 = rnorm(10),
var2 = rnorm(10),
var3 = rnorm(10))

#view data frame
df

var1 var2 var3
1 0.1841698 1.18134622 -0.9410759
2 -1.3535924 -0.73136515 -0.2802438
3 1.0323083 0.06530416 -1.3447057
4 -0.6540649 -0.45005680 1.1222456
5 0.5201189 -0.03688566 -0.6317776
6 0.6119033 -0.13083390 0.7034120
7 -0.1781823 0.56807218 0.2138826
8 -0.1325103 1.10700318 -0.6799447
9 -0.6185180 0.12327017 -0.2411492
10 -0.2699959 -0.04093012 0.5289240
```

Notice that the values in each column of the data frame

are completely different.

Example 2: Generate Random Values Using set.seed()

The following code shows how to use the set.seed() function before using the rnorm() function to create a data frame with three variables that take on random values:

```
#make this example reproducible
set.seed(7)

#create data frame
df <- data.frame(var1 = rnorm(10),
var2 = rnorm(10),
var3 = rnorm(10))

#view data frame
df

var1 var2 var3
1 2.2872472 0.356986230 0.8397504
2 -1.1967717 2.716751783 0.7053418
3 -0.6942925 2.281451926 1.3059647
4 -0.4122930 0.324020540 -1.3879962
5 -0.9706733 1.896067067 1.2729169
```

6 -0.9472799 0.467680511 0.1841928

7 0.7481393 -0.893800723 0.7522799

8 -0.1169552 -0.307328300 0.5917451

9 0.1526576 -0.004822422 -0.9830526

10 2.1899781 0.988164149 -0.2760640

If we use set.seed() with the same seed value as before and create the data frame once again, it's guaranteed to have the same values as the previous data frame:

```
#make this example reproducible
set.seed(7)

#create data frame
df2 <- data.frame(var1 = rnorm(10),
var2 = rnorm(10),
var3 = rnorm(10))

#view data frame
df2
```

var1 var2 var3

1 2.2872472 0.356986230 0.8397504

2 -1.1967717 2.716751783 0.7053418

3 -0.6942925 2.281451926 1.3059647

4 -0.4122930 0.324020540 -1.3879962

5 -0.9706733 1.896067067 1.2729169

6 -0.9472799 0.467680511 0.1841928

7 0.7481393 -0.893800723 0.7522799

8 -0.1169552 -0.307328300 0.5917451

9 0.1526576 -0.004822422 -0.9830526

10 2.1899781 0.988164149 -0.2760640

Notice that the values in this data frame match the ones in the previous data frame.

Note: In this example, we chose to use 7 as our seed value but you can choose whatever number you'd like such as 0, 54, 99, 100, 48787, etc.