What is the difference between ifelse() and if_else() in R?

Authored by

stats writer

June 26, 2024

The ifelse() and if_else() functions in R are conditional statements used for evaluating logical expressions and controlling the flow of a program. While both functions serve a similar purpose, they differ in the way they handle missing values. The ifelse() function returns a vector of the same length as the input vector, with values corresponding to the result of the logical expression. In contrast, the if_else() function preserves the class and attributes of the input vector, and replaces missing values with an error message. Therefore, if_else() is a more strict and precise version of ifelse() that ensures proper handling of missing values. It is important to carefully consider the type of data being used when choosing between these two functions in order to avoid unexpected results.

R: The Difference Between ifelse() vs. if_else()

There are three advantages that the if_else() function in has over the ifelse() function in base R:

1. The if_else() function verifies that both alternatives in the if else statement have the same data type.

2. The if_else() function does not convert Date objects to numeric.

3. The if_else() function offers a 'missing' argument to specify how to handle NA values.

The following examples illustrate these differences in practice.

Example 1: if_else() Verifies that Both Alternatives Have the Same Type

Suppose we have the following data frame in R that contains information about various basketball players:

```
#create data frame
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', 'B', 'B'),
points=c(22, 20, 28, 14, 13, 18, 27, 33))

#view data frame
df

  team points
1    A     22
2    A     20
3    A     28
4    A     14
5    B     13
6    B     18
7    B     27
8    B     33
```

If we use the ifelse() function from base R to create a new column that assigns a value of 'Atlanta' to rows with a team value of 'A' and 0 to rows with a different value, we won't receive any error even though 'Atlanta' is a character and 0 is a number:

```
#create new column based on values in team column
df$city <- ifelse(df$team == 'A', 'Atlanta', 0)

#view updated data frame
df

team points city
1 A 22 Atlanta
2 A 20 Atlanta
3 A 28 Atlanta
4 A 14 Atlanta
5 B 13 0
6 B 18 0
7 B 27 0
8 B 33 0
```

However, if we use the if_else() function from dplyr to perform this same task, we'll receive an error that lets us know we used two different data types in the if else statement:

```
library(dplyr)

#attempt to create new column based on values in team column
```

```
df$city <- if_else(df$team == 'A', 'Atlanta', 0)
```

```
Error: `false` must be a character vector, not a double vector.
```

Example 2: if_else() Does Not Convert Date Objects to Numeric

Suppose we have the following data frame in R that shows the sales made on various dates at some store:

```
#create data frame
df <- data.frame(date=as.Date(c('2022-01-05', '2022-01-17', '2022-01-22',
'2022-01-23', '2022-01-29', '2022-02-13')),
sales=c(22, 35, 24, 20, 16, 19))

#view data frame
df

        date sales
1 2022-01-05    22
2 2022-01-17    35
3 2022-01-22    24
4 2022-01-23    20
5 2022-01-29    16
6 2022-02-13    19
```

If we use the ifelse() function from base R to modify the values in the date column, the values will automatically get converted to numeric:

```
#if date is before 2022-01-20 then add 5 days
df$date <- ifelse(df$date < '2022-01-20', df$date+5, df$date)

date sales
1 19002 22
2 19014 35
3 19014 24
4 19015 20
5 19021 16
6 19036 19

library(dplyr)

#if date is before 2022-01-20 then add 5 days
df$date <- ifelse(df$date < '2022-01-20', df$date+5, df$date)

#view updated data frame
df
```

date sales
1 2022-01-10 22
2 2022-01-22 35
3 2022-01-22 24
4 2022-01-23 20
5 2022-01-29 16
6 2022-02-13 19

Example 3: if_else() Offers a 'missing' Argument to Specify How to Handle NA Values

Suppose we have the following data frame in R:

```
#create data frame
df <- data.frame(team=c('A', 'A', 'A', 'A', 'B', 'B', NA, 'B'),
points=c(22, 20, 28, 14, 13, 18, 27, 33))

#view data frame
df
```

team points
1 A 22
2 A 20
3 A 28
4 A 14

5 B 13
6 B 18
7 <NA> 27
8 B 33

If we use the ifelse() function from base R to create a new column, there is no default option to specify how to handle NA values:

```
#create new column based on values in team column
df$city <- ifelse(df$team == 'A', 'Atlanta', 'Boston')

#view updated data frame
df
```

team points city
1 A 22 Atlanta
2 A 20 Atlanta
3 A 28 Atlanta
4 A 14 Atlanta
5 B 13 Boston
6 B 18 Boston
7 <NA> 27 <NA>
8 B 33 Boston

However, if we use the if_else() function from dplyr then we can use the missing argument to specify how to handle NA values:

```
library(dplyr)

#create new column based on values in team column
df$city <- ifelse(df$team == 'A', 'Atlanta', 'Boston', missing='other')

#view updated data frame
df

team points city
1 A 22 Atlanta
2 A 20 Atlanta
3 A 28 Atlanta
4 A 14 Atlanta
5 B 13 Boston
6 B 18 Boston
7 <NA> 27 other
8 B 33 Boston
```

Notice that the row with an NA value in the team column receives a value of 'other' in the new city column.