R Factors and Factor Levels (With Examples)

Perth Coding Club

R Factors and Factor Levels (With Examples)

In this article, you will learn to work with factors in R programming; a data structure used for predefined, finite number of values. Also, you will learn about levels of a factor.

Factor is a data structure used for fields that takes only predefined, finite number of values (categorical data). For example: a data field such as marital status may contain only values from single, married, separated, divorced, or widowed.

In such case, we know the possible values beforehand and these predefined, distinct values are called levels. Following is an example of factor in R.

```
> x
[1] single  married married single
Levels: married single
```

Here, we can see that factor `x` has four elements and two levels. We can check if a variable is a factor or not using `class()` function.

Similarly, levels of a factor can be checked using the `levels()` function.

```
> class(x)
[1] "factor"
> levels(x)
[1] "married" "single"
```

How to create a factor in R?

We can create a factor using the function `factor()`. Levels of a factor are inferred from the data if not provided.

```
> x <- factor(c("single", "married", "married", "single"));
> x
[1] single  married married single
Levels: married single
> x <- factor(c("single", "married", "married", "single"), levels = c("single", "married", "divorced"));
> x
[1] single  married married single
Levels: single married divorced
```

We can see from the above example that levels may be predefined even if not used.

Factors are closely related with vectors. In fact, factors are stored as integer vectors. This is clearly seen from its structure.

```
> x <- factor(c("single","married","married","single"))
> str(x)
Factor w/ 2 levels "married","single": 2 1 1 2
```

We see that levels are stored in a character vector and the individual elements are actually stored as indices.

Factors are also created when we read non-numerical columns into a data frame.

By default, `data.frame()` function converts character vector into factor. To suppress this behavior, we have to pass the argument `stringsAsFactors = FALSE`.

How to access compoments of a factor?

Accessing components of a factor is very much similar to that of vectors.

```
> x
[1] single  married married single
Levels: married single
> x[3]          # access 3rd element
[1] married
Levels: married single
>  x[c(2, 4)]    # access 2nd and 4th element
[1] married single
Levels: married single
> x[-1]         # access all but 1st element
[1] married married single
Levels: married single
> x[c(TRUE, FALSE, FALSE, TRUE)]  # using logical vector
[1] single single
Levels: married single
```

How to modify a factor?

Components of a factor can be modified using simple assignments. However, we cannot choose values outside of its predefined levels.

```
> x
[1] single  married married single
Levels: single married divorced
> x[2] <- "divorced"   # modify second element;  x
[1] single   divorced married  single
Levels: single married divorced
> x[3] <- "widowed"    # cannot assign values outside levels
Warning message:
In `[<-.factor`(`*tmp*`, 3, value = "widowed") :
invalid factor level, NA generated
> x
[1] single   divorced <NA>     single
Levels: single married divorced
```

A workaround to this is to add the value to the level first.

```
> levels(x) <- c(levels(x), "widowed")    # add new level
> x[3] <- "widowed"
> x
[1] single   divorced widowed  single
Levels: single married divorced widowed
```