**Introduction to factors in R**

In R Language, factors represent categorical variables. Conceptually, categorical variables take a limited number of different values but can be represented by either character or integer values. Understanding of factors in R language is critical to developing statistical modeling because character variables are treated differently in statistical models than continuous variables. By the end of this tutorial on forcats package for working with factors in R, you will be able to inspect levels, change the order of levels, change the values of levels, combine levels, and add/drop levels more efficiently.

But before that, let us learn a bit more about factors in R.

The **first and foremost** thing to remember is that a factor variable in R is represented, or you can say stored as a vector of integer values. Here, each integer represents a character value used to display the levels of character values. You can check that by str() function. When you check the structure of the data frame, you will realize that all the factor variables are denoted by 1, 2, 1 after the colon(:). Let’s take a look for a better understanding.

# Checking structure

str(mtcars)

# Output

'data.frame': 32 obs. of 11 variables:

$ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...

$ cyl : Factor w/ 3 levels "4","6","8": 2 2 1 2 3 2 3 1 1 2 ...

$ disp: num 160 160 108 258 360 ...

$ hp : num 110 110 93 110 175 105 245 62 95 123 ...

$ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...

$ wt : num 2.62 2.88 2.32 3.21 3.44 ...

$ qsec: num 16.5 17 18.6 19.4 17 ...

$ vs : num 0 0 1 1 0 1 0 1 1 1 ...

$ am : num 1 1 1 0 0 0 0 0 0 0 ...

$ gear: num 4 4 4 3 3 3 3 4 4 4 ...

$ carb: num 4 4 1 1 2 1 4 2 2 4 ...

**Second**, both numeric and character variables can be converted to factor variable using as.factor() or factor() function from the forcats package in R.

**Third**, the levels of factors are always stored as character values. You can check the levels of factor variables using levels() function in R.

**Fourth**, factors R can be either **ordered** or **unordered**. Please do not ignore this point; in some analysis or statistical models, the order of the levels may matter.

Now that you are aware of the factors in R. Let’s learn how to execute some of the most frequently used tasks in R involving factors. All the functions mentioned in this tutorial come from forcats package in R. The best part of using forcats package is that it returns tibble, and that means consistency.

**Convert and check levels of factor variables**

As mentioned earlier, here, we will use factor() function to covert cyl variable from mtcars data to factor. We will then check the levels of the variable using levels() function. Finally, we check the class of the variable, which will validate our third point that levels are represented as characters.

1. **Converting an integer variable to factor variable.**

library(forcats)

mtcars$cyl <- factor(mtcars$cyl)

class(mtcars$cyl)

# Output

[1] "factor"

1. **Check levels of a factor variable**

levels(mtcars$cyl)

# Output

[1] "4" "6" "8"

You can see the output values are represented using inverted quotes confirming that levels are stored as character values.

**Inspecting levels of factor variables**

Here we will see how to get the count of each level within a factor using fct\_count(). While we do so, you will also learn how to sort the levels by count using sort= argument. We will then learn how to get the unique values, removing duplicates using fct\_unique() function.

1. **Count the number of values**

fct\_count(mtcars$cyl, sort = TRUE)

# Output

# A tibble: 3 x 2

f n

1 8 14

2 4 11

3 6 7

1. **Remove duplicates to get unique values**

fct\_unique(mtcars$cyl)

# Output

[1] 4 6 8

Levels: 4 6 8

**Changing the order of levels for a factor variable**

There could be multiple reasons for which you would want to change the order of levels in factor variables. As this tutorial is only about R programming language and forcats package. The why and when do we need to order the levels of factor variables is out of scope. However, we will still discuss the different logical approaches one can take to reorder the factor variable levels.

1. **Manually ordering levels of a factor variable**  
   Here the choice is your that is how you wish to reorder the levels. Let’s say you want to reorder levels of cyl variable; then, you can use fct\_relevel() function as illustrated below.

fct\_relevel(mtcars$cyl, c("8", "4", "6")

1. **Reorder factor levels based on the appearance in data**  
   The fct\_inorder() will reorder the levels of a factor variable in R based on the order in which they appear in the data. Below you will notice that 6 appears, then 4 and lastly 8 and so does are factor levels are arranged.

fct\_inorder(mtcars$cyl)

# Output

[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6

[31] 8 4

Levels: 6 4 8

1. **Order factor levels based on the frequency**

The fct\_infreq() function from the forcats package arranges the levels of a factor based on each level’s frequency. The level with the highest frequency takes the first place, followed by lesser frequent levels. It seems most cars in the dataset have 8 cylinders followed by 4 and 6 cylinders.

fct\_infreq(mtcars$cyl)

> fct\_infreq(mtcars$cyl)

[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6

[31] 8 4

Levels: 8 4 6

1. **Reversing the order of levels**  
   If you are interested in reversing the order of the levels of the factors, you can use the fct\_rev() function. You can see we end up with exact reverse order. If you wish, you can check the original order using levels() function.

fct\_rev(mtcars$cyl)

[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6

[31] 8 4

Levels: 8 6 4

1. **Reorder factor levels based on the relationship with other variables**  
   I have not used this function so far, but while I was reading through the documentation of the forcats package, I found this of interest and thus sharing it with you. The example here is borrowed from the project documentation itself. The function is mostly useful for display purposes.

In the example, we order the levels of the color variable based upon its relationship with the a variables. Specifically, we look for minimum values in column a, and you will find that the level blue has the minimum value of 1 and then red has 2 and so on.

The package provides another function called fct\_reorder2(); the function takes into account the relationship with two variables instead of just one.

df <- tibble::tribble(

~color, ~a, ~b,

"blue", 1, 2,

"green", 6, 2,

"purple", 3, 3,

"red", 2, 3,

"yellow", 5, 1

)

df$color <- factor(df$color)

fct\_reorder(df$color, df$a, min)

**Add or drop factor levels in R**

The three functions which are important to know from the addition and deletion perspective are

1. **fct\_expand()** – use it to add new level
2. **fct\_explicit\_na()** – use it, if you wish to assign NA as one of the levels. This way, when you plot charts, NA’s will also appear.
3. **fct\_drop()** – use it drop a particular level

Below we have code snippets with examples for better understanding.

# Adding factor level

fct\_expand(mtcars$cyl,"7")

# Converting NA to factor level

f1 <- factor(c(1, 1, NA, NA,2, 2, NA,2, 1, 2, 2))

f2 <- fct\_explicit\_na(f1, na\_level = "(Unknown)")

# Drop factor level

fac1 <- factor(c("aa","bb"),c("aa","bb","cc"))

fac2 <- fct\_drop(fac1)

fac2

**Changing values of factor levels in R**

The task of changing the levels of variables can be done in multiple ways. One, you may be interested in manual recording. Two, You may be interested in collapsing the levels into lesser groups. Three, You may be interested in clubbing the least/most common levels into a single level. Fourth, You may just want to keep/drop some levels and rename everything as **others**.  
Below is an illustration of how to achieve the above tasks using the functions from the forcats package in R.

1. **Use fct\_collapse() to manually combine levels into defined groups.**

Below we collapse 4 and 6 to form another group called others.

fct\_collapse(mtcars$cyl, Other = c("4", "6"))

1. **Use fct\_other() to replace levels** that you don’t want to keep to others. You can also mention levels that you want to drop; here, the level mentioned in the drop= argument will be named others. The below code produces the exact same results, as mentioned above.

# Example showing keep as argument

fct\_other(mtcars$cyl, keep = c("8"))

# Example showing drop as argument

fct\_other(mtcars$cyl, drop = c("4", "6"))

1. **Use fct\_lump() to group most/least common levels into a single level**. The function is very powerful can provides other statistics to be considered as a measure to combine levels. I encourage you to read more about the function using help(fct\_lump). Below we reserve the most common n values. This again results in the exact same output as mentioned above.

fct\_lump(mtcars$cyl, n = 1)

We also have different variants of the above function.

* fct\_lump\_min(): lumps levels that appear fewer than min times.
* fct\_lump\_prop(): lumps levels that appear in fewer prop \* n times.
* fct\_lump\_n() lumps all levels except for the n most frequent (or least frequent if n < 0)
* fct\_lump\_lowfreq() lumps together the least frequent levels, ensuring that “other” is the smallest.

1. **use fct\_recode() if you wish to replace the values of the levels manually.** The other function which you can use to achieve the same task is fct\_relable().

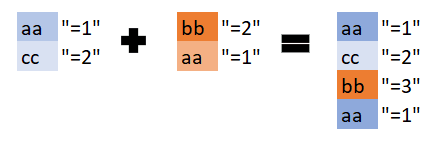
Here we kind of rename the levels of the cyl factor variable to cyl4, cyl6, and cyl8.

fct\_recode(mtcars$cyl, cyl4 = "4", cyl6 = "6", cly8 = "8")

The same task can also be achieved using fct\_relable(). The syntax of the function obeys purrr::map() syntax. The purrr package is an amazing package, and if you have not explored that package yet, I insist that you must.

**Combining factors with different levels**

Often, we get data from different sources, which can also potentially lead to some information available in one source and some in another. For categorical variables, that means that we may now have to patch together factors from these sources because these should have the same levels and not different. The below image illustrates what we mean by factors coming from different sources. This should help you digest the concept.



Example illustrating out after combination of levels from different sources

1. **To combine different levels you can use fct\_c() function.**

# Creating two factors with different levels

fac1 <- factor("aa")

fac2 <- factor("bb")

fct\_c(fac1, fac2)

1. To standardize the factor levels across different sources use fct\_unify(). This is another approach that you can use. Here both lists will have all the levels irrespective of if the value is present or not in the dataset.

# Creating two factors with different levels

fac1 <- factor("aa")

fac2 <- factor("bb")

fct\_unify(list(fac1, fac2))

# Output

[[1]]

[1] aa

Levels: aa bb

[[2]]

[1] bb

Levels: aa bb

Notice how we have both the levels in both the vectors mentioned as part of levels.

With this, we come to an end of the tutorial on the forcats package. I hope you find this tutorial of help and start incorporating some of the functionality discussed here.