Factors (DataCamp Lesson)

The term factor refers to a statistical data type used to store categorical variables. The difference between a categorical variable and a continuous variable is that a categorical variable can belong to a limited number of categories. A continuous variable, on the other hand, can correspond to an infinite number of values.

It is important that R knows whether it is dealing with a continuous or a categorical variable, as the statistical models you will develop in the future treat both types differently. (You will see later why this is the case.)

A good example of a categorical variable is sex. In many circumstances you can limit the sex categories to "Male" or "Female". (Sometimes you may need different categories. For example, you may need to consider chromosomal variation, hermaphroditic animals, or different cultural norms, but you will always have a finite number of categories.)

To create factors in R, you make use of the function factor(). First thing that you have to do is create a vector that contains all the observations that belong to a limited number of categories. For example, `sex_vector` contains the sex of 5 different individuals:

```
sex_vector <- c("Male","Female","Female","Male","Male")
```

It is clear that there are two categories, or in R-terms 'factor levels', at work here: "Male" and "Female".

The function factor() will encode the vector as a factor:

```
factor_sex_vector <- factor(sex_vector)
```

There are two types of categorical variables: a nominal categorical variable and an ordinal categorical variable.

A nominal variable is a categorical variable without an implied order. This means that it is impossible to say that 'one is worth more than the other'. For example, think of the categorical variable $\text{animals}_{\text{vector}}$ with the categories "Elephant", "Giraffe", "Donkey" and "Horse". Here, it is impossible to say that one stands above or below the other. (Note that some of you might disagree ;-)).

In contrast, ordinal variables do have a natural ordering. Consider for example the categorical variable `temperature_vector` with the categories: "Low", "Medium" and "High". Here it is obvious that "Medium" stands above "Low", and "High" stands above "Medium".

When you first get a dataset, you will often notice that it contains factors with specific factor levels. However, sometimes you will want to change the names of these levels for clarity or other reasons. R allows you to do this with the function levels():

```
levels(factor_vector) <- c("name1", "name2",...)
```

A good illustration is the raw data that is provided to you by a survey. A common question for every questionnaire is the sex of the respondent. Here, for simplicity, just two categories were recorded, "M" and "F". (You usually need more categories for survey data; either way, you use a factor to store the categorical data.)

```
survey_vector <- c("M", "F", "F", "M", "M")
```

Recording the sex with the abbreviations "M" and "F" can be convenient if you are collecting data with pen and paper, but it can introduce confusion when analyzing the data. At that point, you will often want to change the factor levels to "Male" and "Female" instead of "M" and "F" for clarity.

Watch out: the order with which you assign the levels is important. If you type `levels(factor_survey_vector)`, you'll see that it outputs [1] "F" "M". If you don't specify the levels of the factor when creating the vector, R will automatically assign them alphabetically. To correctly map "F" to "Female" and "M" to "Male", the levels should be set to c("Female", "Male"), in this order.

After finishing this course, one of your favorite functions in R will be summary(). This will give you a quick overview of the contents of a variable:

```
summary(my_var)
```

Going back to our survey, you would like to know how many "Male" responses you have in your study, and how many "Female" responses. The summary function gives you the answer to this question.

You might wonder what happens when you try to compare elements of a factor. In $\text{factor}_{\text{surveyvector}}$ you have a factor with two levels: "Male" and "Female". But how does R value these relative to each other?

Since "Male" and "Female" are unordered (or nominal) factor levels, R returns a warning message, telling you that the greater than operator is not meaningful. As seen before, R attaches an equal value to the levels for such factors.

But this is not always the case! Sometimes you will also deal with factors that do have a natural ordering between its categories. If this is the case, we have to make sure that we pass this information to R…

Let us say that you are leading a research team of five data analysts and that you want to evaluate their performance. To do this, you track their speed, evaluate each analyst as "slow", "medium" or "fast", and save the results in `speed_vector`.

`speed_vector` should be converted to an ordinal factor since its categories have a natural ordering. By default, the function factor() transforms `speed_vector` into an unordered factor. To create an ordered factor, you have to add two additional arguments: ordered and levels.

```
factor(some_vector,
       ordered = TRUE,
       levels = c("lev1", "lev2" ...))
```

By setting the argument ordered to TRUE in the function factor(), you indicate that the factor is ordered. With the argument levels you give the values of the factor in the correct order.

Having a bad day at work, 'data analyst number two' enters your office and starts complaining that 'data analyst number five' is slowing down the entire project. Since you know that 'data analyst number two' has the reputation of being a smarty-pants, you first decide to check if his statement is true.

The fact that `factor_speed_vector` is now ordered enables us to compare different elements (the data analysts in this case). You can simply do this by using the well-known operators.

Created: 2025-10-20 Mon 09:34