Lesson 07 - Controlling your factors

Robin Donatello

Last Updated 07-16-2019

Introduction

In this lesson we will discuss ways to organize and deal with categorical data, also known as factor data types.

Student Learning Objectives

After completing this lesson students will be able to

- Convert a numeric variable to a factor variable.
- Apply labels to factor
- Understand and control the ordering of the factor.
- Use functions from the `forcats` package.

Prior to this lesson learners should

- Download the [07_factors_notes.Rmd] R markdown file and save into your `scripts/Math130` folder.
- Write the code to import the `email` data set into this notes file. Run it to make sure it works.
- Install the `forcats` package.

```
library(forcats)
email <- read.table("../data/email.txt", header=TRUE, sep="\t")
```

The goal of the `forcats` package is to provide a suite of useful tools that solve common problems with factors. Often in R there are multiple ways to accomplish the same task. Some examples in this lesson will show how to perform a certain task using base R functions, as well as functions from the `forcats` package.

What is a factor?

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 corresponds to a limited number of categories, while a continuous variable can correspond to an infinite number of values.

An example of a categorical variable is the `number` variable in the `email` data set. This variable contains data on whether there was no number, a small number (under 1 million), or a big number in the content of the email.

First we should confirm that R sees `number` as a factor.

```
class(email$number)
```

```
## [1] "factor"
```

We can use the `levels()` function to get to know factor variables.

```
levels(email$number)
```

```
## [1] "big"   "none"  "small"
```

There are three levels: `big`, `none`, and `small`.

How many records are in each level?

Base R

```
table(email$number)
```

```
##
##   big  none small
##   545   549  2827
```

forcats

```
fct_count(email$number)
```

```
## Warning: `list_len()` is deprecated as of rlang 0.2.0.
## Please use `new_list()` instead.
## This warning is displayed once per session.
```

```
## # A tibble: 3 x 2
##   f         n
##   <fct> <int>
## 1 big     545
## 2 none    549
## 3 small  2827
```

Convert a number to Factor

Typically data are entered into the computer using numeric codes such as 0 and 1. These codes stand for categories, such as "no" and "yes". Sometimes we want to analyze these binary variables in two ways:

- For statistical analyses, the data must be numeric 0/1.
- For many graphics, the data must be a factor, "no/yes".

Example: Is the email flagged as spam? The `spam` variable is recorded as an integer variable with values 0 and 1.

```
table(email$spam)
```

```
##
##    0    1
## 3554  367
```

```
class(email$spam)
```

```
## [1] "integer"
```

We use the function `factor()` to convert the numeric variable `spam` to a factor, applying `lables` to convert 0 to "no" and 1 to "yes".

```
email$spam_fac <- factor(email$spam, labels=c("no", "yes"))
```

The ordering of the `labels` argument *must* be in the same order (left to right) as the factor levels themselves. Look back at the order of columns in the `table` - it goes 0 then 1. Thus our labels need to go "no" then "yes".

Always confirm your recode

Here we confirm that the new variable was created corectly by creating a two-way contingency table by calling the `table(old variable, new variable)` function on both the old and new variables.

```
table(email$spam, email$spam_fac, useNA="always")
```

```
##
##          no  yes <NA>
##   0    3554    0    0
##   1       0  367    0
##   <NA>    0    0    0
```

Here we see that all the 0's were recoded to 'no's, and all the 1's recoded to "yes"'s, and there are no new missing values. Success!

Factor ordering

Let's revisit the variable `number`, that contains the size of the number in the email.

```
table(email$number)
```

```
##
##   big  none small
##   545   549  2827
```

Specifically the ordering from left to right of the factors. Seem odd? This is ordinal data, in that `none` is inherently "smaller" than `small`, which is smaller than `big`. But the ordering goes `big` - `none` - `small`, which is in Alphabetical order!

Gee thanks R. Exactly how I want my factors ordered (NOT!) Let's see a few ways of how to control the ordering.

Manually specified

We need to take control of these factors! We can do that by re-factoring the existing factor variable, but this time specifying the `levels` of the factor (since it already has labels).

Base R

```
factor(email$number, levels=c("none", "small", "big")) %>% table()
```

```
## .
##  none small   big
##   549  2827   545
```

forcats

```
email$number %>% fct_relevel("none", "small", "big") %>% table()
```

```
## .
##  none small   big
##   549  2827   545
```

Now it's in a readable, left to right in increasing content size order. This will be important for graphing.

Order by frequency

The size of the number in an email is *ordinal*, meaning the levels have an internal order. *Nominal* categorical data does not have a natural ordering. One preferrable way to order the levels of a nominal variable is by the frequency of the levels. The `forcats` function `fct_infreq()` accomplishes this task.

```
email$number %>% fct_infreq() %>% table()
```

```
## .
## small  none   big
##  2827   549   545
```

There are more emails with small numbers in it than there are emails with no numbers, which shows up more often than emails with big numbers.

Reversed order

Again, `forcats` to the rescue here. Let's remind ourselves what the original ordering was:

```
table(email$number)
```

```
##
##   big  none small
##   545   549  2827
```

And now to reverse this ordering,

```
email$number %>% fct_rev() %>% table()
```

```
## .
## small  none   big
##  2827   549   545
```

This just *happens* to be the same as in decreasing frequency order.

Factor (re)naming

Sometimes factors come to us in names we don't prefer. We want them to say something else.

Base R The easiest way here is to re-factor the variable and apply new `labels`.

```
email$my_new_number <- factor(email$number, labels=c("1M+", "None", "<1M"))
```

Ok, but did this work? Trust, but verify.

```
table(email$number, email$my_new_number, useNA="always")
```

```
##
##          1M+ None  <1M <NA>
##   big    545    0    0    0
##   none     0  549    0    0
##   small    0    0 2827    0
##   <NA>     0    0    0    0
```

The "big" factor is now labeled "1M+", "none" is named "None", and "small" is "<1M".

forcats: use the `fct_recode("NEW" = "old")` function here.

```
email$my_forcats_number <- email$number %>% fct_recode("BIG" = "big", "NONE" = "none", "SMALL" = "small
table(email$number, email$my_forcats_number, useNA="always")
```

```
##
##          BIG NONE SMALL <NA>
##   big    545    0     0    0
##   none     0  549     0    0
##   small    0    0  2827    0
##   <NA>     0    0     0    0
```

Additional resources

- STAT 133 UC Berkeley https://www.stat.berkeley.edu/classes/s133/factors.html
- Be the boss of your factors using `dplyr` and `forcats` http://stat545.com/block029_factors.html
- Wrangling categorical data in R https://peerj.com/preprints/3163/
- The `forcats` vignette can be found at https://forcats.tidyverse.org/
- R for Data Science - chapter on factors https://r4ds.had.co.nz/factors.html