

Working with factor variables in R

Amelia McNamara

June 7, 2016

Introduction

Factors are the data type that R uses for categorical data. Storing categorical data as a factor variable is more efficient than storing the same data as a set of character strings, because factor variables only store the factor labels once.

Factor variables are also important when it comes to modeling, because R will know to consider all cases with the same factor level as belonging to the same group.

So, factors are important. But, they can often be hard to deal with. Because of the way the group numbers are stored separately from the factor labels, it can be easy to overwrite data in such a way that the original data is lost. In this paper, we will consider the best practices for working with factor data.

To do this, we will consider data from the General Social Survey.

Loading the data

We have several options for how to get this data. We could download it in SPSS or Stata formats and use the foreign package to read it in. The GSS download even provides an R file to do the translation for you. Here is the result of that:

```
source('../data/GSS.r')
str(GSS)
```

```
## 'data.frame': 2538 obs. of 17 variables:
## $ YEAR : int 2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 2014 ...
## $ ID_ : int 1 2 3 4 5 6 7 8 9 10 ...
## $ WRKSTAT : int 1 1 4 2 5 1 9 1 8 1 ...
## $ PRESTIGE: int 0 0 0 0 0 0 0 0 0 0 ...
## $ MARITAL : int 3 1 3 1 1 1 1 1 5 1 ...
## $ CHILDS : int 0 0 1 2 3 1 2 2 4 3 ...
## $ AGE : int 53 26 59 56 74 56 63 34 37 30 ...
## $ EDUC : int 16 16 13 16 17 17 12 17 10 15 ...
## $ SEX : int 1 2 1 2 2 2 1 1 2 2 ...
## $ RACE : int 1 1 1 1 1 1 1 1 1 3 ...
## $ INCOM16 : int 2 3 2 2 4 4 2 3 3 1 ...
## $ INCOME : int 12 12 12 12 13 12 13 12 10 12 ...
## $ RINCOME : int 12 12 0 9 0 12 13 12 0 12 ...
## $ INCOME72: int 0 0 0 0 0 0 0 0 0 0 ...
## $ PARTYID : int 5 5 6 5 3 6 6 8 3 3 ...
## $ FINRELA : int 4 4 2 4 3 4 9 3 2 3 ...
## $ SEXORNT : int 3 3 3 3 3 9 0 0 3 3 ...
## - attr(*, "col.label")= chr "Gss year for this respondent" "Respondent id n
```

Obviously, this is less than ideal. Now, all the factor variables are encoded as integers, but their level labels have been lost. We have to look at a codebook to determine if `SEX == 1` indicates male or female. We would rather preserve the integrated level labels. In order to do this, our best option is to download the data as an Excel file and use the `readxl` package to load it.

```
library(readxl)
GSS <- read_excel("../data/GSS.xls")
str(GSS)

## Classes 'tbl_df', 'tbl' and 'data.frame':    2540 obs. of  17 variables:
## $ Gss year for this respondent      : num  2014 2014 2014 2014 2014 ...
## $ Respondent id number              : num   1  2  3  4  5  6  7  8  9 10 ...
## $ Labor force status                : chr  "Working fulltime" "Working fulltime" "
## $ Rs occupational prestige score   (1970) : num   0  0  0  0  0  0  0  0  0  0 ...
## $ Marital status                   : chr  "Divorced" "Married" "Divorced" "Married"
## $ Number of children                : num   0  0  1  2  3  1  2  2  4  3 ...
## $ Age of respondent                 : chr  "53.000000" "26.000000" "59.000000" "56
## $ Highest year of school completed : num  16 16 13 16 17 17 12 17 10 15 ...
## $ Respondents sex                   : chr  "Male" "Female" "Male" "Female" ...
## $ Race of respondent                : chr  "White" "White" "White" "White" ...
## $ Rs family income when 16 yrs old  : chr  "Below average" "Average" "Below average"
## $ Total family income               : chr  "$25000 or more" "$25000 or more" "$2500
## $ Respondents income                : chr  "$25000 or more" "$25000 or more" "Not a
## $ Total family income               : chr  "Not applicable" "Not applicable" "Not a
## $ Political party affiliation        : chr  "Not str republican" "Not str republican"
## $ Opinion of family income          : chr  "Above average" "Above average" "Below a
## $ Sexual orientation                 : chr  "Heterosexual or straight" "Heterosexual"

GSS <- GSS[,-14]
```

That's a little better. Now we have preserved the character strings. But, the data is not yet useable in an analysis.

Renaming the variables

One problem is that the variable names (while human readable) are full of spaces, so are hard to use. But, we can rename them. The `rename()` function in the `dplyr` package is a good way to do this.

```
library(dplyr)

GSS <- GSS %>%
  rename(LaborStatus = `Labor force status`)
```

Considering some factor variables

Once we have variable names that are easier to work with, we can begin to think about how the data should be cleaned.

```
GSS <- GSS %>%
  mutate(LaborStatus = factor(LaborStatus))

levels(GSS$LaborStatus) # I wish I had a piece of dplyr code for this
```

```
## [1] "Keeping house"      "No answer"          "Other"
## [4] "Retired"            "School"              "Temp not working"
## [7] "Unempl, laid off"   "Working fulltime"    "Working parttime"
```

Changing the labels of factors

One action you might want to take is just to change the text of one (or more) of the factor labels, so it appears more nicely formatted in a `ggplot2` plot, for example.

Here is how I do that in base R. Typically, I end up ruining something in the process of doing this, so I *always* start with a summary call, to check after I have done my attempt.

```
summary(GSS$LaborStatus)
```

```
##      Keeping house      No answer      Other      Retired
##             263             2             76            460
##      School Temp not working Unempl, laid off Working fulltime
##             90             40            104           1230
## Working parttime      NA's
##             273             2
```

```
levels(GSS$LaborStatus) <- c(levels(GSS$LaborStatus)[1:5], "Temporarily not working", "Unemployed, laid off", "Unemployed, parttime")
summary(GSS$LaborStatus)
```

```
##      Keeping house      No answer      Other
##             263             2             76
##      Retired      School Temporarily not working
##             460             90             40
## Unemployed, laid off Working full time Working part time
##             104            1230            273
##      NA's
##             2
```

Combining several levels into one

This is another common task. Maybe you want fewer coefficients to interpret in your model, or the process that generated the data makes a finer distinction between categories than your research. For whatever the reason, you want to group together levels that are currently separate.

How I do this in base R:

```
levels(GSS$LaborStatus) <- c("Not employed", "No answer", "Other", "Not employed", "Not employed", "Not employed")
summary(GSS$LaborStatus)
```

```
## Not employed      No answer      Other      Employed      NA's
##             957             2             76            1503             2
```

Combining many categories into one

In this data, age is provided as an integer for respondents 18-88, but then also includes the possible answer “89 or older” as well as a possible “No answer” and NA values.

```
GSS <- GSS %>%
  mutate(`Age of respondent` = factor(`Age of respondent`))
summary(GSS$`Age of respondent`)
```

```
##      18.000000      19.000000      20.000000      21.000000      22.000000      23.000000
##             6             25             26             24             28             30
##      24.000000      25.000000      26.000000      27.000000      28.000000      29.000000
##             31             48             47             41             31             51
```

##	30.000000	31.000000	32.000000	33.000000	34.000000	35.000000
##	57	49	55	47	46	40
##	36.000000	37.000000	38.000000	39.000000	40.000000	41.000000
##	40	54	47	52	46	54
##	42.000000	43.000000	44.000000	45.000000	46.000000	47.000000
##	35	54	39	41	34	43
##	48.000000	49.000000	50.000000	51.000000	52.000000	53.000000
##	32	39	54	45	37	60
##	54.000000	55.000000	56.000000	57.000000	58.000000	59.000000
##	53	52	60	43	60	47
##	60.000000	61.000000	62.000000	63.000000	64.000000	65.000000
##	46	38	44	42	38	40
##	66.000000	67.000000	68.000000	69.000000	70.000000	71.000000
##	35	41	21	23	32	28
##	72.000000	73.000000	74.000000	75.000000	76.000000	77.000000
##	20	22	25	21	24	17
##	78.000000	79.000000	80.000000	81.000000	82.000000	83.000000
##	28	26	16	14	8	11
##	84.000000	85.000000	86.000000	87.000000	88.000000	89 or older
##	13	6	9	8	11	19
##	No answer	NA's				
##	9	2				

We might want to turn this into a factor variable with two levels: 18-65, and over 65. In this case, it would be much easier to deal with a conditional statement about the numeric values, rather than writing out each of the numbers as a character vector.

Acknowledgements

Thanks to my students Kelcie Grenier, Kat Kyuchukov, and Emily Ruppel, whose spring 2016 project in my SDS 291 class formed the inspiration for this paper.