Recoding Variable Values

David Gerard

2019-09-27

Learning Objectives

- Changing the values of a variable based on its current value with `recode()`.
- Changing the values of a variable based on logical statements with `if_else()`.
- Replacing `NA`'s with `replace_na()`.

Change variable values with `recode()`

- Variable values are sometimes uninformative. You might want to change these values before plotting or calculating summary statistics.
- E.g. the estate data in "estate.csv" at https://dcgerard.github.io/stat_412_612/data/estate.csv:

```r
library(tidyverse)
estate <- read_csv(file = "https://dcgerard.github.io/stat_412_612/data/estate.csv")
```

- `estate` contains the following variables:
  - `Price`: Sales price of residence (in dollars)
  - `Area`: Finished area of residence (in square feet)
  - `Bed`: Total number of bedrooms in residence
  - `Bath`: Total number of bathrooms in residence
  - `AC`: `1` = presence of air conditioning, `0` = absence of air conditioning
  - `Garage`: Number of cars that a garage will hold
  - `Pool`: `1` = presence of a pool, `0` = absence of a pool
  - `Year`: Year property was originally constructed
  - `Quality`: Index for quality of construction. `High`, `Medium`, or `Low`.
  - `Style`: Categorical variable indicating architectural style
  - `Lot`: Lot size (in square feet)
  - `Highway`: `1` = highway adjacent, `0` = highway not adjacent.
- It would be better if we could change the `0`/`1` coding for `AC`, `Pool`, and `Highway` to something more informative. That way we won't have to always look up the coding during our analysis.
- `recode()`:
  - Takes a vector as its first argument.
  - Each subsequent argument contains two values separated by an equals sign.
  - The value on the left of the equals sign is the current value inside the vector.
  - The value on the right of the equals sign is the new value for the vector.
  - If the current current value inside the vector is a numeric, then you need to surround its value by backticks "` ` `".
  - It returns a vector with replaced values.
- Toy example:

```
char_vec <- c("a", "a", "b", "c", "c", "a", "b", "b", "c")
recode(char_vec,
       a = "Apple")
```

```
## [1] "Apple" "Apple" "b"     "c"     "c"     "Apple" "b"     "b"     "c"
```

```
recode(char_vec,
       b = "Banana")
```

```
## [1] "a"      "a"      "Banana" "c"      "c"      "a"      "Banana" "Banana"
## [9] "c"
```

```
recode(char_vec,
       a = "Apple",
       b = "Banana",
       c = "Carrot")
```

```
## [1] "Apple"  "Apple"  "Banana" "Carrot" "Carrot" "Apple"  "Banana" "Banana"
## [9] "Carrot"
```

- **Exercise**: In the below vector, recode `"Bob"` to be `"Robert"`, `"John"` to be `"Jonathan"`, and `"Dave"` to be `"David"`.

```
namevec <- c("Bob", "John", "John", "John", "Dave", "Bob", "Bob", "Dave", "John")
```

- Let's use `recode()` in to change the `Quality` values in the `estate` data frame. Recall: we need to use `mutate()` to modify a variable in a data frame.

```
estate %>%
  mutate(Quality = recode(Quality,
                          High   = "Palace",
                          Medium = "Home",
                          Low    = "Slum")) ->
  estate
glimpse(estate)
```

```
## Observations: 522
## Variables: 12
## $ Price   <dbl> 360000, 340000, 250000, 205500, 275500, 248000, 229900...
## $ Area    <dbl> 3032, 2058, 1780, 1638, 2196, 1966, 2216, 1597, 1622, ...
## $ Bed     <dbl> 4, 4, 4, 4, 4, 4, 3, 2, 3, 3, 7, 3, 5, 5, 3, 5, 2, 3, ...
## $ Bath    <dbl> 4, 2, 3, 2, 3, 3, 2, 1, 2, 3, 5, 4, 4, 4, 3, 5, 2, 4, ...
## $ AC      <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, ...
## $ Garage  <dbl> 2, 2, 2, 2, 2, 5, 2, 1, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, ...
## $ Pool    <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ Year    <dbl> 1972, 1976, 1980, 1963, 1968, 1972, 1972, 1955, 1975, ...
## $ Quality <chr> "Home", "Home", "Home", "Home", "Home", "Home", "Home"...
## $ Style   <dbl> 1, 1, 1, 1, 7, 1, 7, 1, 1, 1, 7, 1, 7, 5, 1, 6, 1, 7, ...
## $ Lot     <dbl> 22221, 22912, 21345, 17342, 21786, 18902, 18639, 22112...
## $ Highway <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

- Let's modify `AC`. We'll need backticks here since `1` and `0` are numerics.

```
estate %>%
  mutate(AC = recode(AC,
                     `1` = "AC",
                     `0` = "No AC")) ->
  estate
glimpse(estate)
```

```
## Observations: 522
## Variables: 12
## $ Price   <dbl> 360000, 340000, 250000, 205500, 275500, 248000, 229900...
## $ Area    <dbl> 3032, 2058, 1780, 1638, 2196, 1966, 2216, 1597, 1622, ...
## $ Bed     <dbl> 4, 4, 4, 4, 4, 4, 3, 2, 3, 3, 7, 3, 5, 5, 3, 5, 2, 3, ...
## $ Bath    <dbl> 4, 2, 3, 2, 3, 3, 2, 1, 2, 3, 5, 4, 4, 4, 3, 5, 2, 4, ...
## $ AC      <chr> "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", ...
## $ Garage  <dbl> 2, 2, 2, 2, 2, 5, 2, 1, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, ...
## $ Pool    <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ Year    <dbl> 1972, 1976, 1980, 1963, 1968, 1972, 1972, 1955, 1975, ...
## $ Quality <chr> "Home", "Home", "Home", "Home", "Home", "Home", "Home"...
## $ Style   <dbl> 1, 1, 1, 1, 7, 1, 7, 1, 1, 1, 7, 1, 7, 5, 1, 6, 1, 7, ...
## $ Lot     <dbl> 22221, 22912, 21345, 17342, 21786, 18902, 18639, 22112...
## $ Highway <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

- **Exercise**: Recode the `Highway` and `Pool` variables to have more informative values.

Recode with Logicals with `if_else()`

- Sometimes, it is easier to recode based on logical statements.
- For example, suppose we want to recode the "Bath" variable to have values `1`, `2`, `3`, and `>3`. One way to do this would be:

```
estate %>%
  mutate(Bath = as.character(Bath),
         Bath = recode(Bath,
                       `4` = ">3",
                       `5` = ">3",
                       `6` = ">3",
                       `7` = ">3")) ->
  estate_temp
glimpse(estate_temp)
```

```
## Observations: 522
## Variables: 12
## $ Price   <dbl> 360000, 340000, 250000, 205500, 275500, 248000, 229900...
## $ Area    <dbl> 3032, 2058, 1780, 1638, 2196, 1966, 2216, 1597, 1622, ...
## $ Bed     <dbl> 4, 4, 4, 4, 4, 4, 3, 2, 3, 3, 7, 3, 5, 5, 3, 5, 2, 3, ...
## $ Bath    <chr> ">3", "2", "3", "2", "3", "3", "2", "1", "2", "3", ">3...
## $ AC      <chr> "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", ...
## $ Garage  <dbl> 2, 2, 2, 2, 2, 5, 2, 1, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, ...
## $ Pool    <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ Year    <dbl> 1972, 1976, 1980, 1963, 1968, 1972, 1972, 1955, 1975, ...
## $ Quality <chr> "Home", "Home", "Home", "Home", "Home", "Home", "Home"...
## $ Style   <dbl> 1, 1, 1, 1, 7, 1, 7, 1, 1, 1, 7, 1, 7, 5, 1, 6, 1, 7, ...
## $ Lot     <dbl> 22221, 22912, 21345, 17342, 21786, 18902, 18639, 22112...
## $ Highway <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

- But this is a lot of typing. But we know how to get obtain `TRUE`'s and `FALSE`'s based on whether a house has more than 3 bathrooms.

```r
estate %>%
  mutate(Bath > 3) %>%
  select(contains("Bath")) %>%
  glimpse()
```

```
## Observations: 522
## Variables: 2
## $ Bath       <dbl> 4, 2, 3, 2, 3, 3, 2, 1, 2, 3, 5, 4, 4, 4, 3, 5, 2, ...
## $ `Bath > 3` <lgl> TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FAL...
```

- `if_else()`:
  - Takes a *logical* vector as its first argument.
  - It takes a vector that populates the `TRUE` values as its second argument.
  - It takes a vector that populates the `FALSE` values as its third argument.
  - The second and third arguments *must be the same type* (e.g. both logical, both numeric, both character, etc).
  - The second and third arguments must either be of length 1, or the same length as the logical vector.
  - It returns a vector with replaced values.
- Toy Example:

```r
x <- c(1, 2, 3, 4, 5, 6, 7, 8)
if_else(x > 4, 4, x)
```

```
## [1] 1 2 3 4 4 4 4 4
```

```r
if_else(x > 4, x, 4)
```

```
## [1] 4 4 4 4 5 6 7 8
```

```r
if_else(x > 4, "x > 4", as.character(x))
```

```
## [1] "1"     "2"     "3"     "4"     "x > 4" "x > 4" "x > 4" "x > 4"
```

```r
if_else(x > 4, "x > 4", x) ## should error
```

```
## `false` must be a character vector, not a double vector
```

- **Exercise**: Why did the last `if_else()` call error?
- Let's apply `if_else()` to the `estate` data frame. Recall: we need to use `mutate()` to modify a variable in a data frame.

```r
estate %>%
  mutate(Bath = if_else(Bath > 3,
                        ">3",
                        as.character(Bath))) ->
  estate_temp
glimpse(estate_temp)
```

```
## Observations: 522
## Variables: 12
## $ Price   <dbl> 360000, 340000, 250000, 205500, 275500, 248000, 229900...
## $ Area    <dbl> 3032, 2058, 1780, 1638, 2196, 1966, 2216, 1597, 1622, ...
## $ Bed     <dbl> 4, 4, 4, 4, 4, 4, 3, 2, 3, 3, 7, 3, 5, 5, 3, 5, 2, 3, ...
## $ Bath    <chr> ">3", "2", "3", "2", "3", "3", "2", "1", "2", "3", ">3...
## $ AC      <chr> "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", "AC", ...
## $ Garage  <dbl> 2, 2, 2, 2, 2, 5, 2, 1, 2, 1, 2, 3, 3, 2, 2, 2, 2, 2, ...
## $ Pool    <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ...
## $ Year    <dbl> 1972, 1976, 1980, 1963, 1968, 1972, 1972, 1955, 1975, ...
## $ Quality <chr> "Home", "Home", "Home", "Home", "Home", "Home", "Home"...
## $ Style   <dbl> 1, 1, 1, 1, 7, 1, 7, 1, 1, 1, 7, 1, 7, 5, 1, 6, 1, 7, ...
## $ Lot     <dbl> 22221, 22912, 21345, 17342, 21786, 18902, 18639, 22112...
## $ Highway <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

- **Exercise**: Recode `Price` so that any price less than $250,000 is just listed as `"<250,000"`.

Dealing with `NA`'s by `replace_na()`

- The `starwars` data frame from the dplyr package contains information on different characters from the Star Wars franchise:

```r
data("starwars")
glimpse(starwars)
```

```
## Observations: 87
## Variables: 13
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", ...
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188...
## $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 8...
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "b...
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "l...
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue",...
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0...
## $ gender     <chr> "male", NA, NA, "male", "female", "male", "female",...
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alder...
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human...
## $ films      <list> [<"Revenge of the Sith", "Return of the Jedi", "Th...
## $ vehicles   <list> [<"Snowspeeder", "Imperial Speeder Bike">, <>, <>,...
## $ starships  <list> [<"X-wing", "Imperial shuttle">, <>, <>, "TIE Adva...
```

- The `gender` variable is missing for some individuals

```
starwars %>%
  filter(is.na(gender)) %>%
  select(name, gender)
```

```
## # A tibble: 3 x 2
##   name  gender
##   <chr> <chr>
## 1 C-3PO <NA>
## 2 R2-D2 <NA>
## 3 R5-D4 <NA>
```

- But all of those individuals are droids, so it would be reasonable to replace all of the `NA`s in `gender` with `"droid"`.
- `replace_na()`
  - Takes a vector as its first argument.
  - The second argument is the value with which to replace all `NA`'s.
  - It returns a vector with the `NA`'s replaced.
- Toy example:

```
x <- c("This", "is", "a", NA, NA, "vector")
replace_na(x, "foo")
```

```
## [1] "This"   "is"     "a"      "foo"    "foo"    "vector"
```

- Let's replace the `NA`'s in the `gender` variable in the `starwars` data frame. Recall: we need to use `mutate()` to modify a variable in a data frame.

```
starwars %>%
  mutate(gender = replace_na(gender, "droid")) %>%
  select(name, gender) %>%
  glimpse()
```

```
## Observations: 87
## Variables: 2
## $ name   <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Lei...
## $ gender <chr> "male", "droid", "droid", "male", "female", "male", "fe...
```

- **Exercise**: In the `starwars` data frame, replace the `NA`'s in `hair_color` with `"bald"`.

'