recode variables

Nicholus Tint Zaw

2022-08-20

What we are going to cover in this exercise?

1. ifelse()
2. cut()
3. case_when()

setting dataframe

```
df <- iris
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
summary(df$Sepal.Length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.300   5.100   5.800   5.843   6.400   7.900
```

```
min(df$Sepal.Length)
```

```
## [1] 4.3
```

```
max(df$Sepal.Length)
```

```
## [1] 7.9
```

```
mean(df$Sepal.Length)
```

```
## [1] 5.843333
```

```
median(df$Sepal.Length)
```

```
## [1] 5.8
```

```r
quantile(df$Sepal.Length)
```

```
##   0%  25%  50%  75% 100%
##  4.3  5.1  5.8  6.4  7.9
```

```r
quantile(df$Sepal.Length, probs = 0.25) # 1st quartile
```

```
## 25%
## 5.1
```

```r
quantile(df$Sepal.Length, probs = 0.50) # 2nd quartile
```

```
## 50%
## 5.8
```

```r
quantile(df$Sepal.Length, probs = 0.75) # 3rd quartile
```

```
## 75%
## 6.4
```

ifelse()

```r
# with string variable
df$versicolor <- ifelse(df$Species == "versicolor", "versicolor", "other")

# with numeric variable (>= median = long, < median = short)
df$Sepal.Length.Dummy <- ifelse(df$Sepal.Length >= median(df$Sepal.Length),
                                "long", "short")

# nested ifelse
# (assigned Sepal.Length into three different quantile category)
q1 <- quantile(df$Sepal.Length, probs = 0.25)
q2 <- quantile(df$Sepal.Length, probs = 0.50)
q3 <- quantile(df$Sepal.Length, probs = 0.75)

df$Sepal.Length.Q <- ifelse(df$Sepal.Length < q1,
                            "1st quarter",
                            ifelse(df$Sepal.Length >= q1 & df$Sepal.Length < q2,
                                   "2nd quarter",
                                   ifelse(df$Sepal.Length >= q2 & df$Sepal.Length < q3,
                                          "3rd quarter",
                                          ifelse(df$Sepal.Length >= q3,
                                                 "4th quarter", NA)
                                          )))

table(df$Sepal.Length.Q)
```

```
##
## 1st quarter 2nd quarter 3rd quarter 4th quarter
##          32          41          35          42
```

cut()

```
x <- 1:21

quantile(x)
```

```
##   0%  25%  50%  75% 100%
##    1    6   11   16   21
```

If single vector value was specified at `breaks` parameter, it will equally divided the interested vectors into the number of category specified at `breaks` parameter.

```
cut(x, 3)
```

```
##  [1] (0.98,7.67] (0.98,7.67] (0.98,7.67] (0.98,7.67] (0.98,7.67] (0.98,7.67]
##  [7] (0.98,7.67] (7.67,14.3] (7.67,14.3] (7.67,14.3] (7.67,14.3] (7.67,14.3]
## [13] (7.67,14.3] (7.67,14.3] (14.3,21]   (14.3,21]   (14.3,21]   (14.3,21]
## [19] (14.3,21]   (14.3,21]   (14.3,21]
## Levels: (0.98,7.67] (7.67,14.3] (14.3,21]
```

```
cut(x, 3, dig.lab = 4, ordered = TRUE) # order the levels
```

```
##  [1] (0.98,7.667]  (0.98,7.667]  (0.98,7.667]  (0.98,7.667]  (0.98,7.667]
##  [6] (0.98,7.667]  (0.98,7.667]  (7.667,14.33] (7.667,14.33] (7.667,14.33]
## [11] (7.667,14.33] (7.667,14.33] (7.667,14.33] (7.667,14.33] (14.33,21.02]
## [16] (14.33,21.02] (14.33,21.02] (14.33,21.02] (14.33,21.02] (14.33,21.02]
## [21] (14.33,21.02]
## Levels: (0.98,7.667] < (7.667,14.33] < (14.33,21.02]
```

We can specify the interval criteria by using vector in `breaks` parameter.

```
# check the result! is that what we want?
cut(x, breaks = c(7, 14, 21))
```

```
##  [1] <NA>    <NA>    <NA>    <NA>    <NA>    <NA>    <NA>    (7,14]  (7,14]
## [10] (7,14]  (7,14]  (7,14]  (7,14]  (7,14]  (14,21] (14,21] (14,21] (14,21]
## [19] (14,21] (14,21] (14,21]
## Levels: (7,14] (14,21]
```

```
# revised the syntax a little bit and check the result!
cut(x, breaks = c(1, 7, 14, 21))
```

```
##  [1] <NA>    (1,7]   (1,7]   (1,7]   (1,7]   (1,7]   (1,7]   (7,14]  (7,14]
## [10] (7,14]  (7,14]  (7,14]  (7,14]  (7,14]  (14,21] (14,21] (14,21] (14,21]
## [19] (14,21] (14,21] (14,21]
## Levels: (1,7] (7,14] (14,21]
```

```r
cut(x, breaks = c(0, 7, 14, 21))
```

```
##  [1] (0,7]   (0,7]   (0,7]   (0,7]   (0,7]   (0,7]   (0,7]   (7,14]  (7,14]
## [10] (7,14]  (7,14]  (7,14]  (7,14]  (7,14]  (14,21] (14,21] (14,21] (14,21]
## [19] (14,21] (14,21] (14,21]
## Levels: (0,7] (7,14] (14,21]
```

```r
cut(x, breaks = c(1, 7, 14, 21), include.lowest = TRUE)
```

```
##  [1] [1,7]   [1,7]   [1,7]   [1,7]   [1,7]   [1,7]   [1,7]   (7,14]  (7,14]
## [10] (7,14]  (7,14]  (7,14]  (7,14]  (7,14]  (14,21] (14,21] (14,21] (14,21]
## [19] (14,21] (14,21] (14,21]
## Levels: [1,7] (7,14] (14,21]
```

```r
quantile(x)
```

```
##   0%  25%  50%  75% 100%
##    1    6   11   16   21
```

```r
cut(x, breaks = c(1, 6, 11, 16, 21), include.lowest = TRUE)
```

```
##  [1] [1,6]   [1,6]   [1,6]   [1,6]   [1,6]   [1,6]   (6,11]  (6,11]  (6,11]
## [10] (6,11]  (6,11]  (11,16] (11,16] (11,16] (11,16] (11,16] (16,21] (16,21]
## [19] (16,21] (16,21] (16,21]
## Levels: [1,6] (6,11] (11,16] (16,21]
```

```r
cut(x,
    breaks = c(1, 6, 11, 16, 21),
    include.lowest = TRUE,
    labels = c("1st Q", "2nd Q", "3rd Q", "4th Q"))
```

```
##  [1] 1st Q 1st Q 1st Q 1st Q 1st Q 1st Q 2nd Q 2nd Q 2nd Q 2nd Q 2nd Q 3rd Q
## [13] 3rd Q 3rd Q 3rd Q 3rd Q 4th Q 4th Q 4th Q 4th Q 4th Q
## Levels: 1st Q 2nd Q 3rd Q 4th Q
```

Check the result, when we applied the `right` option parameter. Is it still what we want?

```r
cut(x,
    breaks = c(1, 6, 11, 16, 21),
    include.lowest = TRUE)
```

```
##  [1] [1,6]   [1,6]   [1,6]   [1,6]   [1,6]   [1,6]   (6,11]  (6,11]  (6,11]
## [10] (6,11]  (6,11]  (11,16] (11,16] (11,16] (11,16] (11,16] (16,21] (16,21]
## [19] (16,21] (16,21] (16,21]
## Levels: [1,6] (6,11] (11,16] (16,21]
```

```r
cut(x,
    breaks = c(1, 6, 11, 16, 21),
    include.lowest = TRUE,
    right = FALSE)
```

```
##  [1] [1,6)   [1,6)   [1,6)   [1,6)   [1,6)   [6,11)  [6,11)  [6,11)  [6,11)
## [10] [6,11)  [11,16) [11,16) [11,16) [11,16) [11,16) [16,21] [16,21] [16,21]
## [19] [16,21] [16,21] [16,21]
## Levels: [1,6) [6,11) [11,16) [16,21]
```

If we are not interest about the higher value in the distribution and don't want to specify in the `breaks` parameter vector;

```
x <- 1:100

# only assigned pass/fail using 40 as cut-off point

cut(x,
    breaks = c(0, 40, Inf),
    include.lowest = TRUE,
    labels = c("fail", "pass"))
```

```
##   [1] fail fail fail fail fail fail fail fail fail fail fail fail fail fail fail fail
##  [16] fail fail fail fail fail fail fail fail fail fail fail fail fail fail fail fail
##  [31] fail fail fail fail fail fail fail fail fail fail pass pass pass pass pass pass
##  [46] pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass
##  [61] pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass
##  [76] pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass pass
##  [91] pass pass pass pass pass pass pass pass pass pass
## Levels: fail pass
```

case_when()

We are going to use `case_when()` function from `dplyr` package.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
x <- 1:50

y <-
  case_when(
  x < 5  ~ "< 5",
  x == 7  ~ "equal to 7",
  x > 35  ~ "> 35"
)
```

```r
# check the result and interpret the result!

case_when(
  x %% 35 == 0 ~ "thridy five",
  x %% 5 == 0 ~ "five",
  x %% 7 == 0 ~ "seven",
  TRUE ~ as.character(x)
)
```

```
##  [1] "1"           "2"           "3"           "4"           "five"
##  [6] "6"           "seven"       "8"           "9"           "five"
## [11] "11"          "12"          "13"          "seven"       "five"
## [16] "16"          "17"          "18"          "19"          "five"
## [21] "seven"       "22"          "23"          "24"          "five"
## [26] "26"          "27"          "seven"       "29"          "five"
## [31] "31"          "32"          "33"          "34"          "thridy five"
## [36] "36"          "37"          "38"          "39"          "five"
## [41] "41"          "seven"       "43"          "44"          "five"
## [46] "46"          "47"          "48"          "seven"       "five"
```

```r
# what is %%?
# ref: https://stackoverflow.com/questions/30257819/what-does-the-double-percentage-sign-mean
```