Introduction to R Programming

Factors

Pedro Fonseca

10 May 2020

What are factors?

- A factor is a vector that can contain only predefined values
- Factors are used to represent categorical data

Categorical variables

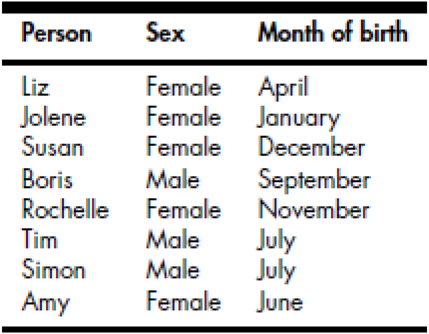
- Categorical variables have a limited and known set of possible outcomes
- Categorical variables typically qualitative

Categorical variables

Examples:

- flip of a coin (heads or tails)
- size of a shirt (S, M, L, XL, XXL, XXXL)
- marital status (single, married, divorced, widowed)
- credit rating (very bad, bad, average, good, excellent)
- school grades (A, B, C, D, E, F)

Categorical variables



Person	Sex	Month of birth
Liz	Female	April
Jolene	Female	January
Susan	Female	December
Boris	Male	September
Rochelle	Female	November
Tim	Male	July
Simon	Male	July
Amy	Female	June

Figure 1: Dataset with categorical variables

More about factors

- Factors are built on top of integers
- They come with two attributes:
  - Levels, which define the set of allowed values
  - Their own class, “factor”, which makes them behave differently from regular integers

Creating a factor

Let's start with a character vector:

```r
blood <- c("B", "AB", "O", "A", "O", "O", "A")

blood
```

```
## [1] "B"  "AB" "O"  "A"  "O"  "O"  "A"
```

Creating a factor

The `factor` function encodes a vector as a factor:

```
blood_factor <- factor(blood)

blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

Note that R sorts the levels alphabetically

Creating a factor

```r
levels(blood_factor)
```

```
## [1] "A"  "AB" "B"  "O"
```

```r
typeof(blood_factor)
```

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

```r
class(blood_factor)
```

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

Creating a factor

When creating a factor, we can set the ordering of the levels:

```
blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

```
blood_factor2 <- factor(blood,
                    levels = c("O", "A", "B", "AB"))
blood_factor2
```

```
## [1] B  AB O  A  O  O  A
## Levels: O A B AB
```

Order levels differently

We can modify the ordering of the levels of an existing factor:

```
blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

```
factor(blood_factor, levels = c("O", "A", "B", "AB"))
```

```
## [1] B  AB O  A  O  O  A
## Levels: O A B AB
```

Order levels differently

`Relevel` re-orders the levels of a factor so that the level specified by `ref` is the first level:

```
blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

```
relevel(blood_factor, ref = "O")
```

```
## [1] B  AB O  A  O  O  A
## Levels: O A AB B
```

Order levels differently

With `rev` we can reverse the ordering of the levels of a factor:

```
blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

```
factor(blood_factor,
  levels = rev(levels(blood_factor)))
```

```
## [1] B  AB O  A  O  O  A
## Levels: O B AB A
```

The internal structure of a factor

The `str` function displays the internal structure of an object:

```
blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

```
str(blood_factor)
```

```
##  Factor w/ 4 levels "A","AB","B","O": 3 2 4 1 4 4 1
```

- Note that the values are stored as integers!
- The levels are just a set of character values to print when the factor is displayed

The internal structure of a factor

```
blood_factor2
```

```
## [1] B  AB O  A  O  O  A
## Levels: O A B AB
```

```
str(blood_factor2)
```

```
##  Factor w/ 4 levels "O","A","B","AB": 3 4 1 2 1 1 2
```

Invalid factor levels

```r
blood_factor
```

```
## [1] B  AB O  A  O  O  A
## Levels: A AB B O
```

```r
blood_factor[3] <- "C"
```

```r
blood_factor
```

```
## [1] B    AB   <NA> A    O    O    A
## Levels: A AB B O
```

Table a factor

How many people there are with each type of blood?

```r
table(blood_factor)
```

```
## blood_factor
##  A AB  B  O
##  2  1  1  2
```

```r
table(blood_factor2)
```

```
## blood_factor2
##  O  A  B AB
##  3  2  1  1
```

Renaming factor levels

```r
blood_factor2
```

```
## [1] B  AB O  A  O  O  A
## Levels: O A B AB
```

```r
levels(blood_factor2) <- c("BT_O", "BT_A", "BT_B",
                           "BT_AB")
blood_factor2
```

```
## [1] BT_B  BT_AB BT_O  BT_A  BT_O  BT_O  BT_A
## Levels: BT_O BT_A BT_B BT_AB
```

Levels and labels

- Levels are input (alphabetic order by default)
- Labels are associated to levels and control how they are displayed in the output

Levels and labels

```
blood <- c("B", "AB", "O", "A", "O", "O", "A", "B")
factor(blood)
```

```
## [1] B  AB O  A  O  O  A  B 
## Levels: A AB B O
```

```
factor(blood,
         labels = c("BT_A", "BT_AB", "BT_B", "BT_O"))
```

```
## [1] BT_B  BT_AB BT_O  BT_A  BT_O  BT_O  BT_A  BT_B 
## Levels: BT_A BT_AB BT_B BT_O
```

Levels and labels

```
factor(blood)
```

```
## [1] B  AB O  A  O  O  A  B
## Levels: A AB B O
```

```
factor(blood,
         levels = c("O", "A", "B", "AB"),
         labels = c("BT_O", "BT_A", "BT_B", "BT_AB"))
```

```
## [1] BT_B  BT_AB BT_O  BT_A  BT_O  BT_O  BT_A  BT_B
## Levels: BT_O BT_A BT_B BT_AB
```

Labels

Duplicated values in `labels` can be used to map different values of the factor to the same level:

```r
factor(blood)
```

```
## [1] B  AB O  A  O  O  A  B
## Levels: A AB B O
```

```r
factor(blood,
         levels = c("O", "A", "B", "AB"),
         labels = c("BT_O", "BT_A", "BT_B", "BT_A"))
```

```
## [1] BT_B BT_A BT_O BT_A BT_O BT_O BT_A BT_B
## Levels: BT_O BT_A BT_B
```

Nominal versus ordinal factors

```r
blood <- c("B", "AB", "O", "A", "O", "O", "A", "B")
blood_factor <- factor(blood)

blood_factor[1] < blood_factor[2]
```

```
## [1] NA
```

This logical comparison is not meaningful, since the factor is not ordered

Nominal versus ordinal factors

Let's build an ordered factor:

```
tshirt <- c("M", "L", "S", "S", "L", "M", "L", "M")

tshirt_factor <- factor(tshirt,
                        ordered = TRUE,
                        levels = c("S", "M", "L"))

tshirt_factor
```

```
## [1] M L S S L M L M
## Levels: S < M < L
```

Nominal versus ordinal factors

```
tshirt_factor[1] < tshirt_factor[2]
```

```
## [1] TRUE
```

```
tshirt_factor[1] > tshirt_factor[2]
```

```
## [1] FALSE
```

Nominal versus ordinal factors

Ordered factors differ from factors in their class:

```
class(blood_factor)
```

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

```
class(tshirt_factor)
```

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

Nominal versus ordinal factors

`ordered(x)` is an alternative to `factor(x, ordered = TRUE)`:

```r
factor(tshirt, ordered = TRUE,
  levels = c("S", "M", "L"))
```

```
## [1] M L S S L M L M
## Levels: S < M < L
```

```r
ordered(tshirt, levels = c("S", "M", "L"))
```

```
## [1] M L S S L M L M
## Levels: S < M < L
```

Testing and coercing

```r
is.factor(blood_factor)
```

```
## [1] TRUE
```

```r
is.ordered(blood_factor)
```

```
## [1] FALSE
```

```r
as.ordered(blood_factor)
```

```
## [1] B  AB O  A  O  O  A  B
## Levels: A < AB < B < O
```

Testing and coercing

```r
is.factor(blood)
```

```
## [1] FALSE
```

```r
as.factor(blood)
```

```
## [1] B  AB O  A  O  O  A  B
## Levels: A AB B O
```

```r
as.ordered(blood)
```

```
## [1] B  AB O  A  O  O  A  B
## Levels: A < AB < B < O
```

Testing and coercing

Curious about the the differences between `factor` and `as.factor`?

- See stackoverflow.com/questions/39279238/why-use-as-factor-instead-of-just-factor

Bivariate tables

```r
hair_color <- factor(c("Brown", "Black", "Black",
                         "Black", "Blond", "Blond",
                         "Black", "Black"))

eye_color <- factor(c( "Blue", "Blue", "Green",
                         "Green", "Green", "Blue",
                         "Blue", "Blue"))
```

Bivariate tables

```
table(hair_color, eye_color)
```

```
##            eye_color
## hair_color Blue Green
##      Black    3     2
##      Blond    1     1
##      Brown    1     0
```

Three-way tables

```r
hair_color <- factor(c("Brown", "Black", "Black",
                         "Black", "Blond", "Blond",
                         "Black", "Black"))

eye_color <- factor(c( "Blue", "Blue", "Green",
                         "Green", "Green", "Blue",
                         "Blue", "Blue"))

shirt_size <- factor(c("L", "S", "S", "M", "L",
                         "L", "S", "S"),
                       ordered = TRUE)
```

Three-way table

```r
table(hair_color, eye_color, shirt_size)
```

Three-way table

```
## , , shirt_size = L
##
##            eye_color
## hair_color Blue Green
##      Black    0     0
##      Blond    1     1
##      Brown    1     0
##
## , , shirt_size = M
##
##            eye_color
## hair_color Blue Green
##      Black    0     1
##      Blond    0     0
##      Brown    0     0
##
## , , shirt_size = S
##
##            eye_color
## hair_color Blue Green
##      Black    3     1
##      Blond    0     0
##      Brown    0     0
```

Creating factors with `cut`

- The `cut` function transforms numerical vectors into factors
- `cut` breaks the range of a numerical vector into intervals
- The limits of the intervals are provided as input

Creating factors with cut

```r
y <- c(5.4, 1.5, 3.33, 0.01, 2, 4.2, 1.99, 1.01)

limits <- c(0, 2, 4, 6)

y_factor <- cut(y, breaks = limits)

y_factor
```

```
## [1] (4,6] (0,2] (2,4] (0,2] (0,2] (4,6] (0,2] (0,2]
## Levels: (0,2] (2,4] (4,6]
```

Creating factors with cut

```
table(y_factor)
```

```
## y_factor
## (0,2] (2,4] (4,6]
##     5     1     2
```

Open and closed intervals

Intervals closed on the right and open on the left:

```r
levels(cut(y, breaks = c(0, 2, 4, 6)))
```

```
## [1] "(0,2]" "(2,4]" "(4,6]"
```

Intervals open on the right and closed on the left:

```r
levels(cut(y, breaks = c(0, 2, 4, 6), right = FALSE))
```

```
## [1] "[0,2)" "[2,4)" "[4,6)"
```

Open and closed intervals

Intervals closed on the right and open on the left, but including the lowest value:

```
levels(cut(y, breaks = c(0, 2, 4, 6),
           include.lowest = TRUE))
```

```
## [1] "[0,2]" "(2,4]" "(4,6]"
```

Intervals open on the right and closed on the left but including the highest value:

```
levels(cut(y, breaks = c(0, 2, 4, 6), right = FALSE,
           include.lowest = TRUE))
```

```
## [1] "[0,2)" "[2,4)" "[4,6]"
```