Week 5: Factor Variables

POP88162 Introduction to Quantitative Research Methods

Tom Paskhalis

Department of Political Science, Trinity College Dublin

So Far

- *Vector* is the core *data structure* in R.
- Vectors can be one of the main data types (*character*, *numeric*, *logical*).
- As opposed to homogeneous vectors, *lists* allow to combine data of different types.
- *Matrices* are vectors with an added class and dimensionality attribute.
- *Data frames* are lists of equal-sized vectors.
- When subsetting data frames the techniques of subsetting matrices and lists are combined.

Topics for Today

- Attributes
- Factors
- Crosstabulation

Review: Data Structures

Structure	Description	Dimensionality	Data Type
`vector`	Atomic vector (scalar)	1d	homogenous
`matrix`	Matrix	2d	homogenous
`array`	One-, two or n-dimensional array	1d/2d/nd	homogenous
`list`	List	1d	heterogeneous
`data.frame`	Rectangular data	2d	heterogeneous

Review: Data Frame Subsetting

- In subsetting data frames the techniques of subsetting matrices and lists are combined
- If you subset with a single vector, it behaves as a list
- If you subset with two vectors, it behaves as a matrix

```
data_frame[row_indices, column_indices]
data_frame[row_indices, column_name(s)]
data_frame[column_indices]
data_frame[column_name(s)]
data_frame$column_name
```

Attributes

- All R objects can have attributes that contain metadata about them.
- Attributes can be thought of as named lists.
- Names, dimensions and class are common examples of attributes.
- They (and some other) have special functions for getting and setting them.

Names Attribute

```r
v <- c(0, 1, 1, 0)
```

```r
# To set names for vector elements we can use names() function
names(v) <- c("Housing Bill", "Pension Bill", "Health Bill", "EU Bill")
v
```

```
Housing Bill Pension Bill  Health Bill      EU Bill
           0            1            1            0
```

```r
# Names of vector elements can be used for subsetting
v[c("Housing Bill", "Health Bill")]
```

```
Housing Bill  Health Bill
           0            1
```

```r
# This is still a numeric vector, don't let the presence of names mislead y
v + 1
```

```
Housing Bill Pension Bill  Health Bill      EU Bill
           1            2            2            1
```

Factors

- Factors form the basis of categorical data analysis in R.
- Values of nominal (categorical) variables represent categories rather than numeric data.
- Internally, in R factor variables are represented by integer vectors.
- With 2 additional attributes:
  - `class()` attribute which is set to `factor`
  - `levels()` attribute which defines allowed values

Factors Example

```
parties <- c("Labour", "Conservative", "Conservative", "Labour", "LibDem")
parties
```

```
[1] "Labour"       "Conservative" "Conservative" "Labour"       "LibDem"
```

```
typeof(parties)
```

```
[1] "character"
```

```
class(parties)
```

```
[1] "character"
```

```
# We use factor() function to convert character vector into factor
# Only unique elements of character vector are considered as a level
parties_factor <- factor(parties)
parties_factor
```

```
[1] Labour       Conservative Conservative Labour       LibDem
Levels: Conservative Labour LibDem
```

Factors Example Continued

```
parties_factor
```

```
[1] Labour       Conservative Conservative Labour       LibDem
Levels: Conservative Labour LibDem
```

```
class(parties_factor)
```

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

```
# Note that the data type of this vector is integer
typeof(parties_factor)
```

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

```
# and not character
is.character(parties_factor)
```

```
[1] FALSE
```

Factors Example Continued

```r
# Note that R automatically sorted the categories alphabetically
levels(parties_factor)
```

```
[1] "Conservative" "Labour"       "LibDem"
```

```r
# You can change the reference category using relevel() function
parties_factor <- relevel(parties_factor, ref = "Labour")
levels(parties_factor)
```

```
[1] "Labour"       "Conservative" "LibDem"
```

```r
# Or define an arbitrary ordering of levels
# using levels argument in factor() function
parties_factor <- factor(
  parties_factor,
  levels = c("Labour", "LibDem", "Conservative")
)
levels(parties_factor)
```

```
[1] "Labour"       "LibDem"       "Conservative"
```

Factors Example Continued

```
1 # Note that there is no meaningful way of converting the original character
2 # into numeric, so NAs are introduced when we try to do so
3 as.numeric(parties)
```

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

```
1 # But this is a perfectly doable operation for factor, even though
2 # we loose some information in the form of level labels
3 as.numeric(parties_factor)
```

```
[1] 1 3 3 1 2
```

Crosstabulation

We can use `table()` function for tabulating a single variable as well as creating contingency tables (crosstabs).

```r
df <- data.frame(
  party = sample(c("left", "right", "centre"), size = 50, replace = TRUE),
  vote = sample(c(0, 1), size = 50, replace = TRUE)
)
```

```r
table(df$party, df$vote)
```

```
        0  1
centre  9 12
left    3 12
right   6  8
```

```r
table(df$vote, df$party)
```

```
  centre left right
0      9    3     6
1     12   12     8
```

Factors in Crosstabs

Implicitly, R treats variables (vectors) that are tabulated as factors.

```r
df$party <- factor(
  df$party,
  levels = c("left", "right", "centre")
)
```

```r
table(df$party, df$vote)
```

```
        0  1
left    3 12
right   6  8
centre  9 12
```

```r
table(df$vote, df$party)
```

```
  left right centre
0    3     6      9
1   12     8     12
```

Next

- Tutorial:
  - Cross Tabulation
- R Assignment 2 due:
  - **08:59 Tuesday, 25 February**
- Next week:
  - Visualisations