Analysing Social Science Data Using R

Class 5: Recapitulation on Vectors and Recoding

Michał Bojanowski[1] mbojan@icm.edu.pl
Zbigniew Karpiński[2] zkarpinski@ifispan.waw.pl

[1]ICM University of Warsaw

[2]IFiS PAN

March 26, 2012
SNS, Warsaw

Three types of vectors

- Numeric
  ```
  > num <- c(1, 3, 25)
  ```
- Character
  ```
  > ch <- c("miner", "driver", "nurse")
  ```
- Logical
  ```
  > lo <- c(TRUE, FALSE, FALSE)
  ```

Vectors with names

```
> names(num) <- ch
> num
```

```
 miner driver  nurse
     1      3     25
```

Subsetting vectors with []

- With numeric vector (by position)

```
> num
 miner driver  nurse
     1      3     25
> num[ c(1,3) ]
miner nurse
    1    25
```

- With character vector (by name)

```
> num[c("driver", "miner")]
driver  miner
     3      1
```

- With logical vector (by logical statement)

```
> num[lo]
miner
```

Factors

"Factor" is a special type of vector with a defined set of admissible values (so-called "levels"), to which optional labels can be assigned. Factors are designed to handle categorical variables.

```
> f <- factor( c(1,2,3,2,1,2,3), levels=1:3,
+         labels=c("a", "b", "c") )
> f
[1] a b c b a b c
Levels: a b c
```

- Labels can be added
- Useful in statistical modeling
  - Dummy variables are constructed automatically.

Values vs levels

```
> f
[1] a b c b a b c
Levels: a b c
> levels(f)
[1] "a" "b" "c"
> levels(f) <- c("minder", "plumber", "driver")
> f
[1] minder  plumber driver  plumber minder  plumber
[7] driver
Levels: minder plumber driver
> f == 1      # does do what we want
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> f == "plumber" # need to refer to levels
[1] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

Replacing with []

Values in vectors can be replaced by assigning new values to selected subsets:

```
> ch[c(3,2)] <- c("fisherman", "plumber")     # element no. 2
> ch
[1] "miner"     "plumber"   "fisherman"
> num["miner"] <- 125          # elements by name
> num
 miner driver  nurse
   125      3     25
> lo[ num < 3 ] <- FALSE   # elements by logical  statement
> lo
[1]  TRUE FALSE FALSE
```

Getting value indices with `which`

Technically `which`, given a logical vector, returns vector of possition of `TRUE`s

```
> x <- c(10, 20, 35, 50, NA, 150)
> x >= 35
[1] FALSE FALSE  TRUE  TRUE    NA  TRUE
> which( x >= 35 )
[1] 3 4 6
```

Using replace function

Using `replace` it is easy to recode into new variables. Basic syntax: `replace( vector, indices, newvalues )`:

```
> x <- c(10, 20, 35, 50, NA, 150)
> replace(x, which(x==10), -1)
[1]  -1  20  35  50  NA 150
> replace(x, which(x == 20 | x == 35), -2)
[1]  10  -2  -2  50  NA 150
> y <- replace(x, which(is.na(x)), 10000)
> y
[1]    10    20    35    50 10000   150
```

The %in% operator

Instead of using compound logical statements, use %in% to test for presence of a set of values

```
> x <- c(10, 20, 35, 50, NA, 150)
> x %in% c(50, 100, 150)
[1] FALSE FALSE FALSE  TRUE FALSE  TRUE
> which(x %in% c(50, 100, 150))
[1] 4 6
```

Helpful in recoding multiple values

```
> replace(x, which(x %in% c(10, 35, 150)), 0)
[1]  0 20  0 50 NA  0
```

Categorizing continuous variables

`cut` continuous variables into intervals by specifying breakpoints:

```
> x <- c(-1, 1, 1, 2, 3, 4, 4, 5, 6, 7, 8, 9, 10)
> b <- cut(x, c(-Inf, 3, 5, 7, Inf))
> b
 [1] (-Inf,3] (-Inf,3] (-Inf,3] (-Inf,3] (-Inf,3]
 [6] (3,5]    (3,5]    (3,5]    (5,7]    (5,7]
[11] (7, Inf] (7, Inf] (7, Inf]
Levels: (-Inf,3] (3,5] (5,7] (7, Inf]
```

Rank order

Functions `order` and `rank`.

```
> x <- c(2,2,2,1,1,1)
> y <- c(3,2,1,1,2,3)
> order(x, y)
[1] 4 5 6 3 2 1
```

`rank` provides smarter ways of dealing with ties

```
> rank(x)    # ties.method="average"
[1] 5 5 5 2 2 2
> rank(x, ties.method="first")
[1] 4 5 6 1 2 3
> rank(x, ties.method="min")
[1] 4 4 4 1 1 1
> rank(x, ties.method="max")
[1] 6 6 6 3 3 3
```