chapter 4

Basic data management

Instructor: Li, Han

Contents

- Creat and recode variables
- Rename the variables
- Handle missing values
- Convert data type
- Handle date value
- Sort variables

As the author said:

“Data is a messy business — a very, very messy business.” In my own work, as much as 60% of the time I spend on data analysis is focused on preparing the data for analysis. ”

A working example

This example is studying how men and women differ in their ways they lead the organizations. The questions q1-q5 are asking their bosses to rate their deferential behavior.

This manager asks my opinion before making personnel decisions.

1	2	3	4	5
strongly disagree	disagree	neither agree nor disagree	agree	strongly agree

Table 4.1 Gender differences in leadership behavior

Manager	Date	Country	Gender	Age	q1	q2	q3	q4	q5
1	10/24/08	US	M	32	5	4	5	5	5
2	10/28/08	US	F	45	3	5	2	5	5
3	10/01/08	UK	F	25	3	5	5	5	2
4	10/12/08	UK	M	39	3	3	4		
5	05/01/09	UK	F	99	2	2	1	2	1

Creat the data frame

Listing 4.1 Creating the leadership data frame

```
manager <- c(1, 2, 3, 4, 5)
date <- c("10/24/08", "10/28/08", "10/1/08", "10/12/08", "5/1/09")
country <- c("US", "US", "UK", "UK", "UK")
gender <- c("M", "F", "F", "M", "F")
age <- c(32, 45, 25, 39, 99)
q1 <- c(5, 3, 3, 3, 2)
q2 <- c(4, 5, 5, 3, 2)
q3 <- c(5, 2, 5, 4, 1)
q4 <- c(5, 5, 5, NA, 2)
q5 <- c(5, 5, 2, NA, 1)
leadership <- data.frame(manager, date, country, gender, age,
                         q1, q2, q3, q4, q5, stringsAsFactors=FALSE)
```

Interesting questions

- What is the mean rating scores of each manager?
- In surveys, respondents often skip questions. How to handle the incomplete data?
- we may only be interested in a few variables. To simplify matters, how to create a new dataset with only the variables of interest.

- Leadership behavior may change as a function of the manager’s age. We may recode the current values of age into a new categorical age grouping (for example, young, middle-aged, elder).
- Leadership behavior may change over time. We may limit the study to data gathered during a specific period of time (say, January 1, 2009 to December 31, 2009).

Creat new variables

We could create new variables and transform existing ones.

```
dataframe$variable <- expression
```

Example 1

```
mydata<-data.frame(x1 = c(2, 2, 6, 4),
x2 = c(3, 4, 2, 8))

mydata$sumx  <-  mydata$x1 + mydata$x2
mydata$meanx <- (mydata$x1 +    mydata$x2)/2
```

Table 4.2 Arithmetic operators

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
^ or **	Exponentiation
x%%y	Modulus (x mod y) 5%%2 is 1
x%/%y	Integer division 5%/%2 is 2

transform()

The above codes are equivalent to

```
mydata <- transform(mydata, sumx  =  x1 + x2,
meanx = (x1 + x2)/2)
```

Recode variables

Recode variables conditional on their existing values, for example,

- Change a continuous variable into a set of categories
- Replace miscoded values with correct values
- Create a pass/fail variable based on a set of cutoff scores

Table 4.3 Logical operators

Operator	Description
`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Exactly equal to
`!=`	Not equal to
`!x`	Not *x*
`x | y`	*x* or *y*
`x & y`	*x* and *y*
`isTRUE(x)`	Test if *x* is `TRUE`

Example 2

```
leadership$age[leadership$age  == 99] <- NA

leadership$agecat[leadership$age > 75]   <- "Elder"
leadership$agecat[leadership$age >= 55 &
        leadership$age <= 75]   <- "Middle Aged"
leadership$agecat[leadership$age  < 55] <- "Young"
```

RGui (32-bit) - [R Console]

File Edit View Misc Packages Windows Help

```
  manager     date country gender age q1 q2 q3 q4 q5
1       1 10/24/08      US      M  32  5  4  5  5  5
2       2 10/28/08      US      F  45  3  5  2  5  5
3       3  10/1/08      UK      F  25  3  5  5  5  2
4       4 10/12/08      UK      M  39  3  3  4 NA NA
5       5   5/1/09      UK      F  99  2  2  1  2  1
> leadership$age[leadership$age  == 99] <- NA
> leadership$agecat[leadership$age > 75]   <- "Elder"
> leadership$agecat[leadership$age >= 55 &
+ leadership$age <= 75]   <- "Middle Aged"
> leadership$agecat[leadership$age  < 55]  <- "Young"
> leadership
  manager     date country gender age q1 q2 q3 q4 q5 agecat
1       1 10/24/08      US      M  32  5  4  5  5  5  Young
2       2 10/28/08      US      F  45  3  5  2  5  5  Young
3       3  10/1/08      UK      F  25  3  5  5  5  2  Young
4       4 10/12/08      UK      M  39  3  3  4 NA NA  Young
5       5   5/1/09      UK      F  NA  2  2  1  2  1   <NA>
>
```

The above codes are equivalent to

```
leadership <- within(leadership,{
    agecat <- NA
    agecat[age > 75]  <- "Elder"
    agecat[age >= 55 & age <= 75] <- "Middle Aged"
    agecat[age < 55]  <- "Young" })
```

Rename variables

We could change the names of the variables to new ones.

Example 3

```
names(leadership)
names(leadership)[2] <- "testDate"
names(leadership)[6:10] <- c("item1", "item2", "item3", "item4", "item5")
leadership
```

Missing values

In a survey, data is likely to be incomplete. In R, missing values are represented by the symbol NA (not available). Impossible values (for example, dividing by 0) are represented by the symbol NaN (not a number).

The function is.na() allows you to check whether there is any missing value.

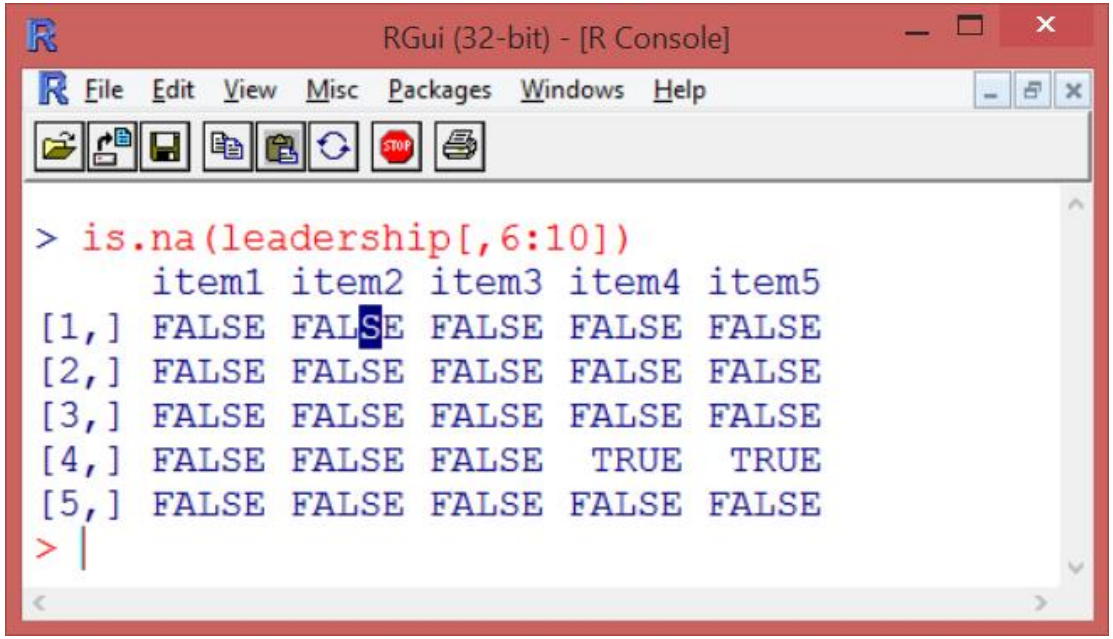
Example 4

```
y <- c(1, 2, 3, NA)
is.na(y)
```

Then R returns c(FALSE, FALSE, FALSE, TRUE)

Example 5

is.na(leadership[,6:10])



Recode values to be missing

Example 6

```
leadership$age[leadership$age  == 99] <- NA
```

Exclude missing values from analysis

Example 7

```
x <- c(1, 2, NA, 3)
y <- sum(x, na.rm=TRUE)
```

na.omit()

na.omit() will delete the rows with missing data.

Example 8

```
newdata <- na.omit(leadership)
newdata
```

Date value

Dates are typically entered into R as character strings and then translated into date variables using as.Date().

Table 4.4 Date formats

Symbol	Meaning	Example
%d	Day as a number (0–31)	01–31
%a	Abbreviated weekday	Mon
%A	Unabbreviated weekday	Monday
%m	Month (00–12)	00–12

Table 4.4 Date formats *(continued)*

Symbol	Meaning	Example
`%b`	Abbreviated month	Jan
`%B`	Unabbreviated month	January
`%y`	2-digit year	07
`%Y`	4-digit year	2007

The default format for inputting dates is yyyy-mm-dd.

Example 9

```
mydates <- as.Date(c("2007-06-22", "2004-02-13"))

strDates <- c("01/05/1965", "08/16/1975")
dates <- as.Date(strDates, "%m/%d/%Y")
```

Example 10

```
myformat <- "%m/%d/%y"

leadership$date <- as.Date(leadership$testDate,
myformat)
```

R can tell you what date it is today!

Example 11

```
Sys.Date()
date()
```

Type conversions

R provides a set of functions to identify an object’s data type and convert it to a different data type.

is.datatype(x) returns TRUE or FALSE.

x <- as.datatype(x) changes the data type of x.

Table 4.5 Type conversion functions

Test	Convert
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`
`is.factor()`	`as.factor()`
`is.logical()`	`as.logical()`

Example 12

```
a <- c(1,2,3)
is.numeric(a)
is.vector(a)

a <- as.character(a)
a
is.numeric(a)
is.vector(a)
is.character(a)
```

Sort data

To sort a data frame in R, use the order() function. By default, the sorting order is ascending.

Sorting the variable with a minus sign to indicate a descending order.

Example 12

```
newdata <- leadership[order(leadership$age),]

attach(leadership)
newdata <- leadership[order(gender, age), ]
detach(leadership)

attach(leadership)
newdata <-leadership[order(gender, -age), ]
detach(leadership)
```

Summary

In this session, we have learned

- Creat and recode variables
- Rename the variables
- Handle missing values
- Convert data type
- Handle date value
- Sort variables

Try the examples and have fun!