Managing Data

MS

9/15/2021

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com. A cheat sheet can be found at https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf.

Reference: Code in this document comes from R in Action, Second Edition

---

An Example

```
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
```

Questions

- The five ratings (q1 to q5) need to be combined, yielding a single mean deferential score from each manager.
- In surveys, respondents often skip questions. For example, the boss rating manager 4 skipped questions 4 and 5. You need a method of handling incomplete data. You also need to recode values like 99 for age to missing.
- There may be hundreds of variables in a dataset, but you may only be interested in a few. To simplify matters, you'll want to create a new dataset with only the variables of interest.
- Past research suggests that leadership behavior may change as a function of the manager's age. To examine this, you may want to recode the current values of age into a new categorical age grouping (for example, young, middle-aged, elder).
- Leadership behavior may change over time. You might want to focus on deferential behavior during the recent global financial crisis. To do so, you may want to limit the study to data gathered during a specific period of time (say, January 1, 2009 to December 31, 2009)

Creating New Variables

New variables can simply be created as *variable <- expression* The following code provides three ways of creating new variables, such that the new variables are also incorporated into the original data frame.

```r
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

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

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

Recoding Variables

Recoding involves creating new values for variables based on some conditions. For example, you might want to

- Change a continuous variable into a category
- Replace missing values
- Create a variable based on cutoff values

To recode, we use R's logical operators.

Assume you want to recode the ages of the managers in the leadership dataset from continuous values to a categorical variable *agecat (Young, Middle Aged, Elder)*. You need to also recode the value 99 as missing.

```r
leadership$age[leadership$age == 99] <- NA # This recodes the variable age, when age is 99 to NA
# The statement *variable[condition]<-expression will only make the assignment when TRUE

# Now we can create the categorical variable
leadership$agecat[leadership$age > 75] <- "Elder"
leadership$agecat[leadership$age >= 55 &  leadership$age <= 75] <- "Middle Aged"
leadership$agecat[leadership$age < 55] <- "Young"

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

# agecat is now a character variable. We should really turn it into an ordered factor.
leadership$agecat <- factor(leadership$agecat,ordered = TRUE,levels=c("Elder","Middle Aged","Young"))
```

Renaming Variables

```
fix(leadership) # To interactively rename variables, or
names(leadership)[2] <- "testDate" # renames the second variable to *testDate*
```

Missing Values

The function is.na() allows you to test for the presence of missing values

```
is.na(leadership$q4)
```

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

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

```
##      q1    q2    q3    q4    q5
## 1 FALSE FALSE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE FALSE FALSE
## 3 FALSE FALSE FALSE FALSE FALSE
## 4 FALSE FALSE FALSE  TRUE  TRUE
## 5 FALSE FALSE FALSE FALSE FALSE
```

```
leadership$age[leadership$age == 99] <- NA # This codes values of 99 to NA, so they are not included in

# We might also want to exclude missing values from analyses. For example
z <- sum(leadership$q4, na.rm = T)

# We can also remove any observation with missing data using the na.omit() function.

leadership
```

```
##   manager testDate 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>
```

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

```
##   manager testDate 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
```

Type Conversions

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()

```r
a <- c(1,2,3)
a
```

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

```r
is.numeric(a)
```

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

```r
is.vector(a)
```

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

```r
a <- as.character(a)
a
```

```
## [1] "1" "2" "3"
```

```r
is.numeric(a)
```

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

```r
is.vector(a)
```

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

```r
is.character(a)
```

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

Sorting and Merging

```r
newdata <- leadership[order(leadership$age),] # sorts by age
newdatad <- leadership[order(-leadership$age),] # sorts by descending order of age

attach(leadership)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     age, country, gender, manager, q1, q2, q3, q4, q5
```

```r
newdata <-leadership[order(gender, -age),] # sort by gender, and descending age
detach(leadership)
```

To merge two data frames (datasets) horizontally, you use the merge() function. In most cases, two data frames are joined by one or more common key variables (that is, an inner join). For example,

> total <- merge(dataframeA, dataframeB, by="ID")

merges dataframeA and dataframeB by ID. Similarly,

> total <- merge(dataframeA, dataframeB, by=c("ID","Country"))

If you're joining two matrices or data frames horizontally and don't need to specify a common key, you can use the cbind() function:

> total <- cbind(A, B)

To join two data frames (datasets) vertically, use the rbind() function:

> total <- rbind(dataframeA, dataframeB)