Transformations and Recoding with Open Source R

Revolution Analytics

1 Introduction

2 R Functions and Operators

3 Creating and Changing Variables

4 Conditional Transformation and Recoding

5 Renaming Variables

Overview

- In this module we will discuss simple variable transformations

Outline

1 Introduction

2 R Functions and Operators

3 Creating and Changing Variables

4 Conditional Transformation and Recoding

5 Renaming Variables

Introduction

- Variable transformations can be performed easily in R through the use of built-in and user-defined functions
- We can easily create variables using objects in the R workspace

Outline

1 Introduction

2 R Functions and Operators

3 Creating and Changing Variables

4 Conditional Transformation and Recoding

5 Renaming Variables

R Functions

- R functions take arguments and create output values after performing the steps and operations defined in its body.
- For our purposes we will focus on functions useful for transforming (mostly) numeric data

R Operators

- R operators for computation (+, −, /, ∗, etc.) can be used to transform data and variables directly without defining a function
- The same applies for R functions.

Outline

1 Introduction

2 R Functions and Operators

3 Creating and Changing Variables

4 Conditional Transformation and Recoding

5 Renaming Variables

Creating and Changing Variables

- Let's use the pre-loaded `mtcars` dataset in the the `datasets` package, containing data for motor trend car road tests
- We will make a copy called `mymtcars` so that we can keep track of our new variables

```
mymtcars <- mtcars
str(mymtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
...
```

Creating and Changing Variables

What if we needed to create the following variables?

1. `wt2` defined as the weight of the car in pounds (instead of thousand pounds as in `wt`)
2. `HpPerThouPound` defined as the amount of horsepower per unit weight
3. `RaRSqr` defined as the square of the rear axle ratio `drat`

Operators

```
mymtcars$wt2 <- mymtcars[["wt"]] * 1000
mymtcars$HpPerThouPound <- mymtcars[["hp"]]/mymtcars$wt
mymtcars$RaRSqr <- mymtcars$drat^2
```

View the Dataset

```
str(mymtcars)
```

```
## 'data.frame':    32 obs. of  14 variables:
##  $ mpg          : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl          : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp         : num  160 160 108 258 360 ...
##  $ hp           : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat         : num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt           : num  2.62 2.88 2.32 3.21 3.44 ...
...
```

```
head(mymtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb  wt2
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 2620
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 2875
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 2320
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 3215
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 3440
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 3460
...
```

Using Functions

```
mymtcars$logdisp <- log(mymtcars$disp)
head(mymtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb  wt2
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 2620
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 2875
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 2320
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 3215
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 3440
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 3460
...
```

Upshot

If at all possible, we want to operate on entire **variables** with each operation or function call, not **instances**

This approach is called vectorized arithmetic, and there are many benefits:

```
- Faster execution: Vectorized functions in R are frequent
- Easier to Read and Debug: Single statements are expressi
```

Exercise

Let's take a moment and practice with the `cars` dataset. This dataset includes initial velocities and stopping distances for a set of cars. Based on this information, let's compute average acceleration for each row.

$$v_{avg} = \frac{v_f + v_i}{2}$$

$$t = \frac{d}{v_{avg}}$$

$$a = \frac{v_f - v_i}{t}$$

Creating and Changing Variables

- Changing Variables works in the same manner, we only need to assign the transformation to the same column or variable
- For example, we could change the scale of the `disp` variable from in^3 to *cc*

```
mymtcars$disp <- mymtcars$disp * 2.54^3
```

Creating Categorical Variables

factor()	Simple categorical variable creation, where each unique value is converted to its own level
cut()	More complex variable creation, where you specify the range of each level

Categorical Variable Examples

```
mymtcars$cylFact <- factor(mymtcars$cyl)
qbreaks <- quantile(mymtcars$qsec)
qbreaks[1] <- qbreaks[1] - 0.01
mymtcars$qsecCut <- cut(mymtcars$qsec, qbreaks)
summary(mymtcars$qsecCut)
```

```
## (14.5,16.9] (16.9,17.7] (17.7,18.9] (18.9,22.9]
##           8           8           9           7
```

Outline

1 Introduction

2 R Functions and Operators

3 Creating and Changing Variables

4 Conditional Transformation and Recoding

5 Renaming Variables

Conditional Transformation or Recoding

- In many cases, we wouldn’t want to execute the same transformation for all entries or observations in the dataset
- Suppose we only want to do the weight transformation if the car is automatic and leave the other weights untouched

Example

```
mymtcars$wt[mymtcars$am == 0] <- mymtcars$wt[mymtcars$am == 0] * 1000
head(mymtcars)
```

```
##                    mpg cyl     disp  hp drat       wt  qsec vs am gear
## Mazda RX4         21.0   6 2621.930 110 3.90    2.620 16.46  0  1    4
## Mazda RX4 Wag     21.0   6 2621.930 110 3.90    2.875 17.02  0  1    4
## Datsun 710        22.8   4 1769.803  93 3.85    2.320 18.61  1  1    4
## Hornet 4 Drive    21.4   6 4227.863 110 3.08 3215.000 19.44  1  0    3
## Hornet Sportabout 18.7   8 5899.343 175 3.15 3440.000 17.02  0  0    3
## Valiant           18.1   6 3687.089 105 2.76 3460.000 20.22  1  0    3
...
```

Inefficient Example

- This is equivalent to an inefficient loop:

```
for (i in 1:nrow(mymtcars)) {
  if (mymtcars$am[i] == 0) {
    mymtcars$wt[i] <- mymtcars$wt[i] * 1000
  }
}
```

Questions?

Outline

1 Introduction

2 R Functions and Operators

3 Creating and Changing Variables

4 Conditional Transformation and Recoding

5 Renaming Variables

Renaming Variables

- Renaming variables or data also falls under basic transformations. In R we simply edit the `names` (or some variant) attribute of the object for this purpose

```
names(mymtcars)

##  [1] "mpg"           "cyl"           "disp"          "hp"
##  [5] "drat"          "wt"            "qsec"          "vs"
##  [9] "am"            "gear"          "carb"          "wt2"
## [13] "HpPerThouPound" "RaRSqr"        "logdisp"       "cylFact"
## [17] "qsecCut"

names(mymtcars)[2:4] <- c("cylinder", "displacement", "horsepower")
names(mymtcars)

##  [1] "mpg"           "cylinder"      "displacement"  "horsepower"
##  [5] "drat"          "wt"            "qsec"          "vs"
##  [9] "am"            "gear"          "carb"          "wt2"
## [13] "HpPerThouPound" "RaRSqr"        "logdisp"       "cylFact"
## [17] "qsecCut"
```

Renaming Variables

- An alternative is to use the `rename` function in the `reshape` package

```
library(reshape)
mymtcars <- rename(mymtcars, c(displacement = "D", horsepower = "H"))
```

Questions?

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR