Introduction to “R”
Data manipulation

Arndt Leininger

@a_leininger
arndt.leininger@fu-berlin.de

28 September 2019

Reading files and loading packages

Reading Data

Some things to pay attention to when reading data

- Keep an original copy of the data exactly as you found it, if you make changes save to a new name
- Try and make changes with R in a script. If you have to switch to Excel at least write down what you did. ("Friends don't let friends use Excel for data analysis")
- Check the data has been imported properly before you use it.
  - You may need to specify what character signifies missing values in your data
  - You might need to specify the delimiter in a `.csv` file

Reading a file from another statistical package

- The package `foreign` provides functions to read datasets saved in other statistical software
- The function we'll use is `read.dta()`
- The package `foreign` is included in every installation of R but is not loaded when starting R
- First we have to load the foreign package

Loading packages

- R comes with a number of build-in packages and thousands of community contributed packages which extend its functionality
- You can load installed libraries with the `library()` function

```
library("foreign")
```

```
library(foreign)
```

- With the library function there's no need to put the package name in quotation marks

Reading a (Stata) file

```r
df <- read.dta("data/auto.dta")
```

```r
dim(df)
```

```
## [1] 74 12
```

```r
names(df)
```

```
##  [1] "make"         "price"        "mpg"
##  [4] "rep78"        "headroom"     "trunk"
##  [7] "weight"       "length"       "turn"
## [10] "displacement" "gear_ratio"   "foreign"
```

Data manipulation

`data.frames`

- A `data.frame` is a matrix which allows (column) vectors to be of different types
- A `matrix` can be thought of as a collection of column or row vectors

Vectors

- Any variable is a vector
- A `data.frame` is simply a collection of vectors of the same length (bot not necessarily type)

```
make <- df$make

make
```

```
##  [1] "AMC Concord"       "AMC Pacer"
##  [3] "AMC Spirit"        "Buick Century"
##  [5] "Buick Electra"     "Buick LeSabre"
##  [7] "Buick Opel"        "Buick Regal"
##  [9] "Buick Riviera"     "Buick Skylark"
## [11] "Cad. Deville"      "Cad. Eldorado"
## [13] "Cad. Seville"      "Chev. Chevette"
## [15] "Chev. Impala"      "Chev. Malibu"
## [17] "Chev. Monte Carlo" "Chev. Monza"
## [19] "Chev. Nova"        "Dodge Colt"
## [21] "Dodge Diplomat"    "Dodge Magnum"
## [23] "Dodge St. Regis"   "Ford Fiesta"
```

Acccessing elements of a vector

```r
# First element of a vector
make[1]
```

```
## [1] "AMC Concord"
```

```r
# Second element of a vector
df$make[2]
```

```
## [1] "AMC Pacer"
```

```r
# Elements three to five
make[3:5]
```

```
## [1] "AMC Spirit"    "Buick Century" "Buick Electra"
```

```r
# Elements three and five
df$make[c(1, 3)]
```

```
## [1] "AMC Concord" "AMC Spirit"
```

Acccessing elements of a vector

```r
price <- as.integer(c(222000000, 120000000, 105000000))

names(price) <- c('Neymar', 'Coutinho', 'Dembélé')

price['Neymar']
```

```
##    Neymar
## 222000000
```

```r
price
```

```
##    Neymar  Coutinho   Dembélé
## 222000000 120000000 105000000
```

Enter the matrix

- A `data.frame` is a matrix allowing (column) vectors to be of different types
- A `matrix` can be thought of as a collection of column or row vectors

```r
# A matrix

m <- matrix(data = c("1,1", "1,2", "1, ...", "1, k", "2,1",
    "2,2", "2, ...", "2, k", "...,1", "..., 2", "...", "..., k",
    "n, 1", "n, 2", "n, ...", "n,k"), byrow = T, ncol = 4)
m
```

```
##      [,1]    [,2]     [,3]     [,4]
## [1,] "1,1"   "1,2"    "1, ..." "1, k"
## [2,] "2,1"   "2,2"    "2, ..." "2, k"
## [3,] "...,1" "..., 2" "..."    "..., k"
## [4,] "n, 1"  "n, 2"   "n, ..." "n,k"
```

Enter the matrix

- These can be accessed by stating the row or column index

```
# e.g.
m[1, ]  # the first row
```

```
## [1] "1,1"    "1,2"    "1, ..." "1, k"
```

```
m[, 2]  # the second column
```

```
## [1] "1,2"    "2,2"    "..., 2" "n, 2"
```

Matrices

- A rectangle of dimension n (number of rows) and k (number of columns)
- Any element of a matrix can be accessed by reference to its row and column position
- `matrixname[row number(s), column name(s) or number(s)]`

```
# e.g.
m[1, 2]
```

```
## [1] "1,2"
```

```
m[2, -3]
```

```
## [1] "2,1"  "2,2"  "2, k"
```

```
m[5, 2]
```

```
## Error in m[5, 2]: subscript out of bounds
```

Subsetting

- `data.frames` can be subset column-wise based on column number(s) or name(s)

```
df[, 1]
```

```
##  [1] "AMC Concord"       "AMC Pacer"
##  [3] "AMC Spirit"        "Buick Century"
##  [5] "Buick Electra"     "Buick LeSabre"
##  [7] "Buick Opel"        "Buick Regal"
##  [9] "Buick Riviera"     "Buick Skylark"
## [11] "Cad. Deville"      "Cad. Eldorado"
## [13] "Cad. Seville"      "Chev. Chevette"
## [15] "Chev. Impala"      "Chev. Malibu"
## [17] "Chev. Monte Carlo" "Chev. Monza"
## [19] "Chev. Nova"        "Dodge Colt"
## [21] "Dodge Diplomat"    "Dodge Magnum"
## [23] "Dodge St. Regis"   "Ford Fiesta"
## [25] "Ford Mustang"      "Linc. Continental"
## [27] "Linc. Mark V"      "Linc. Versailles"
## [29] "Merc. Bobcat"      "Merc. Cougar"
```

Subsetting

- `data.frames` can be subset column-wise based on column number(s) or name(s)

```
df[, "make"]
```

```
##  [1] "AMC Concord"       "AMC Pacer"
##  [3] "AMC Spirit"        "Buick Century"
##  [5] "Buick Electra"     "Buick LeSabre"
##  [7] "Buick Opel"        "Buick Regal"
##  [9] "Buick Riviera"     "Buick Skylark"
## [11] "Cad. Deville"      "Cad. Eldorado"
## [13] "Cad. Seville"      "Chev. Chevette"
## [15] "Chev. Impala"      "Chev. Malibu"
## [17] "Chev. Monte Carlo" "Chev. Monza"
## [19] "Chev. Nova"        "Dodge Colt"
## [21] "Dodge Diplomat"    "Dodge Magnum"
## [23] "Dodge St. Regis"   "Ford Fiesta"
## [25] "Ford Mustang"      "Linc. Continental"
## [27] "Linc. Mark V"      "Linc. Versailles"
## [29] "Merc. Bobcat"      "Merc. Cougar"
```

Subsetting

- `data.frames` can be subset column-wise based on column number(s) or name(s)

```
df[, c(1, 3)]
```

```
##                  make mpg
## 1         AMC Concord  22
## 2           AMC Pacer  17
## 3          AMC Spirit  22
## 4       Buick Century  20
## 5       Buick Electra  15
## 6       Buick LeSabre  18
## 7          Buick Opel  26
## 8         Buick Regal  20
## 9       Buick Riviera  16
## 10      Buick Skylark  19
## 11       Cad. Deville  14
## 12      Cad. Eldorado  14
## 13       Cad. Seville  21
## 14     Chev. Chevette  29
```

Subsetting

- `data.frames` can be subset column-wise based on column number(s) or name(s)

```
df[, c("make", "mpg")]
```

```
##                  make mpg
## 1         AMC Concord  22
## 2           AMC Pacer  17
## 3          AMC Spirit  22
## 4       Buick Century  20
## 5       Buick Electra  15
## 6       Buick LeSabre  18
## 7          Buick Opel  26
## 8         Buick Regal  20
## 9       Buick Riviera  16
## 10      Buick Skylark  19
## 11       Cad. Deville  14
## 12      Cad. Eldorado  14
## 13       Cad. Seville  21
## 14     Chev. Chevette  29
```

Subsetting

- `data.frames` can be subset row-wise based on row number(s) or boolean statements
- Suppose you want to know what the most expensive car is

```
df[which(df$price == max(df$price)), "make"]
```

```
## [1] "Cad. Seville"
```

What is `which()`?

- `which()` takes a logical vector as input and returns the index values for `TRUE`

```
d <- data.frame(x = c(1, 2, 3), y = c(21, 18, NA))

d[d$y == 18, ]
```

```
##     x  y
## 2   2 18
## NA NA NA
```

```
d[which(d$y == 18), ]
```

```
##   x  y
## 2 2 18
```

- ignoring `NA`s

Subsetting

Say you want to subset the data to domestic cars

```
df2 <- df[which(df$foreign == "Domestic"), ]
df2
```

```
##                  make price mpg rep78 headroom trunk
## 1         AMC Concord  4099  22     3      2.5    11
## 2           AMC Pacer  4749  17     3      3.0    11
## 3          AMC Spirit  3799  22    NA      3.0    12
## 4       Buick Century  4816  20     3      4.5    16
## 5       Buick Electra  7827  15     4      4.0    20
## 6       Buick LeSabre  5788  18     3      4.0    21
## 7          Buick Opel  4453  26    NA      3.0    10
## 8         Buick Regal  5189  20     3      2.0    16
## 9       Buick Riviera 10372  16     3      3.5    17
## 10      Buick Skylark  4082  19     3      3.5    13
## 11       Cad. Deville 11385  14     3      4.0    20
## 12      Cad. Eldorado 14500  14     2      3.5    16
## 13       Cad. Seville 15906  21     3      3.0    13
## 14     Chev. Chevette  3299  29     3      2.5     9
```

Subsetting

The average price of domestic cars is also simple to obtain

```
mean(df[which(df$foreign == "Domestic"), "price"])
```

```
## [1] 6072.423
```

Subsetting

Subset to expensive domestic cars, i.e. cars that are "Domestic" and cost more than 10000

```
df2 <- df[which(df$foreign == "Domestic" & df$price > 10000),
    ]
```

- & means AND
- | means OR

Digression: & and |

What are the results of the following?

1. T & T =
2. T & F =
3. T | T =
4. T | F =

Digression: & and |

What are the results of the following?

1. $\mathsf{T} \mathbin{\&} \mathsf{T} = \mathsf{T}$
2. $\mathsf{T} \mathbin{\&} \mathsf{F} = \mathsf{F}$
3. $\mathsf{T} \mid \mathsf{T} = \mathsf{T}$
4. $\mathsf{T} \mid \mathsf{F} = \mathsf{T}$

Digression: >, <, <=, >=, ==, !=

`x > y` If x is larger than y return `TRUE`.

`x < y` If x is smaller than y return `TRUE`.

`x <= y` If x is smaller or equal to y return `TRUE`.

`x >= y` If x is larger or equal to y return `TRUE`.

`x == y` If x is equal to y return `TRUE`.

`x != y` If x is unequal to y return `TRUE`.

Recoding variables

- Create a dummy variable indicating a domestic car

```
df$domestic <- FALSE
df$domestic[which(df$foreign == "Domestic")] <- TRUE
```

- Alternatively

```
df$domestic <- df$foreign == "Domestic"
```

- The value TRUE is copied into all observations of the variable for which `foreign == 'Domestic'`
- Note the double equation sign ==
- Note the quotation signs around 'Domestic'

Recoding variables

- Create a variable which contains the price of domestic cars

```
df$price_domestic <- as.numeric(NA)

df$price_domestic[which(df$domestic == TRUE)] <- df$price
```

```
## Warning in df$price_domestic[which(df$domestic ==
## TRUE)] <- df$price: number of items to replace is not a
## multiple of replacement length
```

```
df$price_domestic[which(df$domestic == TRUE)] <- df$price[which(
    TRUE)]
```

- The vector on the RHS needs to be shorter or equal to the length of the vector on the LHS

Why stringsAsFactor = F?

```
df <- read.csv('data/BundestagForecastReplicationData.csv')
df$chancellor[15:16] <- 'Gerhard Schröder'
```

```
## Warning in `[<-.factor`(`*tmp*`, 15:16, value =
## structure(c(1L, 6L, 6L, : invalid factor level, NA
## generated
```

```
df <- read.csv('data/BundestagForecastReplicationData.csv',
               stringsAsFactors = F)
df$chancellor[15:16] <- 'Gerhard Schröder'
```

Saving and loading objects

- R allows you to save any kind of object, not just datasets to the hard drive.

```
a <- 1
b <- c(T, F)

save(a, b, file = "data/data.RData")
```

```
a <- 1
b <- c(T, F)

save(a, b, file = "../data/data.RData")
```

Saving and loading objects

```r
load("data/data.RData")

a
b
```

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

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

- Load puts the objects `a` and `b` into R's memory

Hands-on I

Hands-on I

`hands-on/02_datamanipulation/hands-on1.R`