5 Subsetting

(AST230) R for Data Science

Md Rasel Biswas

Subsetting

- R's subsetting operators are fast and powerful and mastering them allows you concisely perform complex operations
- Subsetting in R easy to learn but hard to master because you need to internalize a number of interrelated concepts
  - There are three subsetting operators, `[[`, `[`, and `$`
  - subsetting operators interact differently with different vector types (e.g. atomic vectors, lists, factors, matrices, and data frames)
  - Subsetting can be combined with assignment

Subsetting atomic vectors

- We have already discussed how to select elements from an atomic vector using numerical and logical indexing previously

```
age <- c(11, 9, 8, 10, 5)
age[c(2, 3, 5)]
```

```
[1] 9 8 5
```

```
age[-c(1, 3)]
```

```
[1]  9 10  5
```

```
age[age >= 8 & age <= 10]
```

```
[1]  9  8 10
```

Subsetting matrices

```
mat <- matrix(1:9, nrow = 3)
mat
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Select specific element of a matrix

```
mat[1, 3]
```

```
[1] 7
```

```
mat[1:2, 3]
```

```
[1] 7 8
```

Select rows or columns of a matrix

```
mat[ , 2:3]
```

```
     [,1] [,2]
[1,]    4    7
[2,]    5    8
[3,]    6    9
```

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

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    3    6    9
```

Subsetting matrices

```
mat
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
mat[1, ] # returns a vector
```

```
[1] 1 4 7
```

```
mat[, 1] # returns a vector
```

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

```
# returns a matrix
mat[1, , drop = FALSE]
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
```

```
# returns a matrix
mat[, 1, drop = FALSE]
```

```
     [,1]
[1,]    1
[2,]    2
[3,]    3
```

Subsetting data frames

```
# Creating a data frame
df <- data.frame(x = 1:4, y = letters[1:4], z = 11:14)
```

```
df
```

```
  x y  z
1 1 a 11
2 2 b 12
3 3 c 13
4 4 d 14
```

```
# Variable names
names(df)
```

```
[1] "x" "y" "z"
```

```
str(df)
```

```
'data.frame':   4 obs. of  3 variables:
 $ x: int  1 2 3 4
 $ y: chr  "a" "b" "c" "d"
 $ z: int  11 12 13 14
```

```
# Row names
row.names(df)
```

```
[1] "1" "2" "3" "4"
```

Subsetting data frames

Positional Indexing

```
df[1:2, 2:3]
```

```
  y  z
1 a 11
2 b 12
```

```
df[2:3, ] # not specifying column index means we want all columns
```

```
  x y  z
2 2 b 12
3 3 c 13
```

```
df[ ,1:2] # similar
```

```
  x y
1 1 a
2 2 b
3 3 c
4 4 d
```

Subsetting data frames

Extract a specific variable `x` from data frame

```
# first method
df$x
```

```
[1] 1 2 3 4
```

```
# second method
df[[1]]
```

```
[1] 1 2 3 4
```

```
# third method
df[["x"]]
```

```
[1] 1 2 3 4
```

```
# fourth method
df["x"]
```

```
  x
1 1
2 2
3 3
4 4
```

Subsetting data frames

```
names(df)
```

```
[1] "x" "y" "z"
```

```
# Select the variables x and y
df[, c("x", "y")]
```

```
  x y
1 1 a
2 2 b
3 3 c
4 4 d
```

```
# Select the variables x and y
df[c("x", "y")]
```

```
  x y
1 1 a
2 2 b
3 3 c
4 4 d
```

```
# Selecting both columns and rows
df[1:2, c("x", "y")]
```

```
  x y
1 1 a
2 2 b
```

Subsetting data frames

Logical Indexing

```
df
```

```
  x y  z
1 1 a 11
2 2 b 12
3 3 c 13
4 4 d 14
```

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

```
  x y  z
1 1 a 11
3 3 c 13
```

```
df$x > 2
```

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

```
# Rows that satisfy x>2
df[df$x > 2, ]
```

```
  x y  z
3 3 c 13
4 4 d 14
```

Exercise 5

- How many variables are in `mtcars`? Show the list of these variables.
- Extract the vector `mpg` from `mtcars`, and calculate its mean and standard deviation.
- Check whether there is any missing value in `wt` of `mtcars`
- Obtain a data frame with `mpg > 22`
- Obtain a data frame from `mtcars` with `gear=5` and `cyl=4` and keep only the variables `year`, `model`, and `cyl`

Subsetting lists

- `list()` is the most flexible data structure of R, vectors of different lengths and/or a data frame can be included in a list
- Data frame is a special case of a list and `[[ ]]` is useful for extracting elements of a list
- List is considered as a heterogeneous vector as its elements could be of different types
- The operators `[[`, `[`, and `$` can be used to selecting elements from a list

Subsetting lists

Create a list

```
my_list <- list(1:3, "a", c(TRUE, FALSE, FALSE), c(2L, 5L, 9L))
```

```
str(my_list)
```

```
List of 4
 $ : int [1:3] 1 2 3
 $ : chr "a"
 $ : logi [1:3] TRUE FALSE FALSE
 $ : int [1:3] 2 5 9
```

```
my_list[[1]]
```

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

```
my_list[1]
```

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

```
typeof(my_list[[1]])
```

```
[1] "integer"
```

```
typeof(my_list[1])
```

```
[1] "list"
```

Subsetting lists

```
l2 <- list(x = 1:5,
           y = c(TRUE, FALSE),
           z = matrix(1:4, 2))
l2
```

```
$x
[1] 1 2 3 4 5

$y
[1]  TRUE FALSE

$z
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

Extracting x

```
l2$x
```

```
[1] 1 2 3 4 5
```

```
l2[["x"]]
```

```
[1] 1 2 3 4 5
```

```
l2[[1]]
```

```
[1] 1 2 3 4 5
```