Intro to R

Ibrahim O Alabi, PhDc

2023-06-10

What is subsetting in programming?

Subsetting is a fundamental operation in R that allows you to extract specific elements, rows, or columns from data structures such as vectors, matrices, or data frames. The criteria for subsetting can vary and may involve conditions based on values, positions, logical operations, or other expressions.

By the end of this tutorial, you'll have a good understanding of different subsetting techniques and be able to apply them to various scenarios. Let's get started!

Subsetting Vectors

In R, vectors are one-dimensional arrays. You can subset vectors by specifying the position or indices of the elements you want to extract. There are two common methods for subsetting vectors:

1. **Basic Indexing**: You can use square brackets `[ ]` and specify the indices or positions of the elements you want to extract. For example:

```
# creating a vector
my_vector <- c(10, 20, 30, 40, 50)

# Access the second element
my_vector[2]
```

```
## [1] 20
```

```
# Access elements 3 to 5
my_vector[3:5]
```

```
## [1] 30 40 50
```

```
# Access specific elements
my_vector[c(2,5)]
```

```
## [1] 20 50
```

2. **Logical Indexing**: You can create a logical condition that evaluates to `TRUE` or `FALSE` for each element of the vector. By using this condition within square brackets, you can extract only the elements that satisfy the condition. For example:

- Subsetting a vector based on a logical condition

```
# Create a vector
my_vector <- c(10, 20, 30, 40, 50, 26, 36, 73, 2, 205, 33, 85)

# Subsetting the vector based on a logical condition (greater than 30)
condition <- my_vector>30
```

```r
sub_set<-my_vector[condition]
sub_set
```

```
## [1]  40  50  36  73 205  33  85
```

Explanation: In this example, the code creates a numeric vector called `my_vector`. It then defines a logical condition by comparing each element of `my_vector` with 30, resulting in a logical vector condition that indicates which elements satisfy the condition (TRUE for elements greater than 30, and FALSE for elements less than or equal to 30). Finally, the code subsets `my_vector` using the logical condition, extracting only the elements that correspond to TRUE in the `condition` vector. The resulting subset is stored in a new vector called `sub_set`, which contains the elements of `my_vector` that are greater than 30.

- Subsetting a vector based on multiple logical conditions

```r
# Subsetting the vector based on multiple logical conditions (even numbers less than 40)
condition<-my_vector %% 2 ==0 & my_vector < 40
sub_set <-my_vector[condition]
sub_set
```

```
## [1] 10 20 30 26 36  2
```

Explanation: In this example, we use multiple logical conditions to subset the vector `my_vector`. The conditions are that the elements should be even (divisible by 2) and less than 40.

- Subsetting a character vector based on a logical condition

```r
# Create a character vector
string_of_names <- c("John", "Jane", "Alice", "Bob")

# Subsetting the vector based on a logical condition (names starting with "J")
condition <-substr(string_of_names, 1, 1) == 'J'
sub_set <- string_of_names[condition]
sub_set
```

```
## [1] "John" "Jane"
```

Explanation: In this example, we have a character vector names that contains names. We subset the vector based on a logical condition that selects names starting with the letter "J". The resulting subset contains the names John, Jane.

- Subsetting a vector based on a logical condition using the `which()` function

```r
# Subsetting the vector using the `which()` function (greater than 30)
condition <- which(my_vector > 30)
sub_set <- my_vector[condition]
sub_set
```

```
## [1]  40  50  36  73 205  33  85
```

Explanation: In this example, the `which()` function is used to obtain the indices of elements in the vector `my_vector` that satisfy the condition of being greater than 30. These indices are then used for logical indexing to subset the vector.

- Subsetting a vector based on negation of a logical condition

```r
# Create a vector
numbers <- 1:10

# Subsetting the vector based on the negation of a logical condition (not divisible by 3)
sub_set <- numbers[!(numbers %% 3 == 0)]
sub_set
```

```
## [1]  1  2  4  5  7  8 10
```

Explanation: In this example, the vector numbers contains `numbers` from 1 to 10. We use the negation operator (`!`) to subset the vector by selecting only the elements that are not divisible by 3.

- Subsetting a character vector based on a pattern

```
# Subsetting the vector based on a pattern (names starting with "J")
condition<-grep("^J", string_of_names)
sub_set <- string_of_names[condition]
sub_set
```

```
## [1] "John" "Jane"
```

Explanation: In this example, the `grep()` function is used to search for a specific pattern, which is names starting with "J", within the vector `string_of_names`. The function returns the indices of the matching elements, which are then used for indexing to subset the vector.

- Subsetting a factor vector based on levels

```
# Create a factor vector
grades <- factor(c("A", "B", "C", "A", "B"), levels = c("A", "B", "C", "D", "E"))

# Subsetting the vector based on levels (select "A" and "B" grades)
condition<-grades %in% c("A", "B")
sub_set <- grades[condition]
sub_set
```

```
## [1] A B A B
## Levels: A B C D E
```

Explanation: In this example, the `%in%` operator is used to check if the elements of the factor vector `grades` are present in the specified levels ("A" and "B"). The result is a logical vector that indicates whether each element matches the specified levels. This logical vector is then used for indexing to subset the vector.

These examples demonstrate various applications of logical indexing for subsetting vectors in R. You can combine logical conditions, use the `which()` function to obtain indices, work with logical vectors, and use negation to create subsets based on specific criteria or conditions.