Assessment 06 - Indexing

Alessandro Corradini - Harvard Data Science Professional

Logical Vectors

Here we will be using logical operators to create a logical vector.

Instructions

- Compute the per 100,000 murder rate for each state and store it in an object called `murder_rate`.
- Then use the logical operators to create a logical vector, name it `low`, that tells us which entries of murder_rate are lower than 1.

```
library(dslabs)
data(murders)

# Store the murder rate per 100,000 for each state, in `murder_rate`
murder_rate <- murders$total / murders$population * 100000

# Store the `murder_rate < 1` in `low`
low <- murder_rate < 1
```

which

The function `which()` help us know directly, which values are low or high, etc. Let's use it in this question.

Instructions

Use the results from the previous exercise and the function `which` to determine the indices of `murder_rate` associated with values lower than 1.

```
# Store the murder rate per 100,000 for each state, in murder_rate
murder_rate <- murders$total/murders$population*100000

# Store the murder_rate < 1 in low
low <- murder_rate < 1

# Get the indices of entries that are below 1
which(low)
```

```
##  [1] 12 13 16 20 24 30 35 38 42 45 46 51
```

Ordering vectors

Note that if we want to know which entries of a vector are lower than some values we can use code like this:

```
small <- murders$population < 1000000
murders$state[small]
```

The code above shows us the states with populations smaller than one million.

Instructions

Use the results from the previous exercise to report the names of the states with murder rates lower than 1, using the square brackets to retrieve the name of the states from the dataset.

```
# Store the murder rate per 100,000 for each state, in murder_rate
murder_rate <- murders$total/murders$population*100000
```

```
# Store the murder_rate < 1 in low
low <- murder_rate < 1

# Names of states with murder rates lower than 1
murders$state[low]
```

```
##  [1] "Hawaii"        "Idaho"         "Iowa"          "Maine"
##  [5] "Minnesota"     "New Hampshire" "North Dakota"  "Oregon"
##  [9] "South Dakota"  "Utah"          "Vermont"       "Wyoming"
```

Filtering

Instructions

Now we will extend the code from the previous exercises to report the states in the Northeast with murder rates lower than 1.

- Define `low` as before.
- Use the `&` operator to create a new object `ind` that is true when `low` is true and the state is in the Northeast
- Use the brackets `[` and `ind` to show the state names that satisfy this condition

```
# Store the murder rate per 100,000 for each state, in `murder_rate`
murder_rate <- murders$total/murders$population*100000

# Store the `murder_rate < 1` in `low`
low <- murder_rate < 1

# Create a vector ind for states in the Northeast and with murder rates lower than 1.
ind <- low & murders$region == "Northeast"
# Names of states in `ind`
murders$state[ind]
```

```
## [1] "Maine"         "New Hampshire" "Vermont"
```

Filtering continued

Instructions

In a previous exercise we computed the murder rate for each state and the average of these numbers.

How many states are below the average?

```
# Store the murder rate per 100,000 for each state, in murder_rate
murder_rate <- murders$total/murders$population*100000

# Compute average murder rate and store in avg using `mean`
avg <- mean(murder_rate)

# How many states have murder rates below avg ? Check using sum
sum(murder_rate < avg)
```

```
## [1] 27
```

Match

In this exercise we use the `match` function to identify the states with abbreviations AK, MI, and IA.

Instructions

Define a character vector with the abbreviations. Start by defining an index of the entries of `murders$abb` that match the three abbreviations. Use the `[` operator to extract the states.

```
# Store the 3 abbreviations in abbs in a vector (remember that they are character vectors and need quot
abbs <- c('AK', 'MI', 'IA')
# Match the abbs to the murders$abb and store in ind
ind <- match(abbs, murders$abb)
# Print state names from ind
murders$state[ind]
```

```
## [1] "Alaska"   "Michigan" "Iowa"
```

%in%

If rather than an index we want a logical that tells us whether or not each element of a first vector is in we can use the function `%in%`. For example:

```
x <- c(2, 3, 5)
y <- c(1, 2, 3, 4)
x%in%y
```

Gives us a two `TRUE` followed by a `FALSE` because 2 and 3 are in `y`.

Instructions

Which of the following are actual abbreviations: MA, ME, MI, MO, MU?

- Define a character vector with the abbreviations MA, ME, MI, MO, MU.
- Use the `%` operator to create a logical vectors that is TRUE when the abbreviation is in `murders$abb`.

```
# Store the 5 abbreviations in `abbs`. (remember that they are character vectors)
abbs <- c('MA', 'ME', 'MI', 'MO', 'MU')

# Use the %in% command to check if the entries of abbs are abbreviations in the the murders data frame
abbs %in% murders$abb
```

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

Logical operator

Instructions

We are again working with the characters `abbs <- c("MA", "ME", "MI", "MO", "MU")`

- In a previous exercise we computed the index `abbs%in%murders$abb`. Use `which` and the `!` operator to get the index of the entries of abbs that are not abbreviations.
- Show the entries of `abbs` that are not actual abbreviations.

```
# Store the 5 abbreviations in abbs. (remember that they are character vectors)
abbs <- c("MA", "ME", "MI", "MO", "MU")

# Use the `which` command and `!` operator to find out which abbreviation are not actually part of the
ind <- which(!abbs%in%murders$abb)
# What are the entries of abbs that are not actual abbreviations
abbs[ind]
```

```
## [1] "MU"
```