Indexing

```
library(dslabs)
```

```
## Warning: package 'dslabs' was built under R version 4.1.3
```

```
data(murders)
```

R provides a powerful and convenient way of indexing vectors. We can, for example, subset a vector based on properties of another vector. We continue our US murders example to demonstrate.

Subsetting with logicals

We can calculate the murder rate using

```
murder_rate <- murders$total / murders$population * 100000
```

Say you are moving from Italy where, according to an ABC news report, the murder rate is only 0.71 per 100,000. You would prefer to move to a state with a similar rate. Another powerful feature of R is that we can we can use logicals to index vectors. Note that if we compare a vector to a single number, it actually performs the test for each entry. Here is an example related to the question above.

```
ind <- murder_rate < 0.71
ind
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [49] FALSE FALSE FALSE
```

```
ind2 <- murder_rate <= 0.71 & murder_rate>0.2
```

Or if we want to know if its less than or equal to we can use

```
ind <- murder_rate <= 0.71
ind
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [49] FALSE FALSE FALSE
```

Note that we get back a logical vector with `TRUE` for each entry smaller than or equal to 0.71. To see which states these are, we can leverage the fact that vectors can be indexed with logicals.

```
murders$state[ind]
```

```
## [1] "Hawaii"        "Iowa"          "New Hampshire" "North Dakota"
## [5] "Vermont"
```

Note that to count how many are TRUE, the function `sum` returns the sum of the entries of a vector and logical vectors get *coerced* to numeric with `TRUE` coded as 1 and `FALSE` as 0. Thus we can count the states using

```
sum(ind)
```

```
## [1] 5
```

Logical Operators

Suppose we like the mountains and we want to move to a safe state in the West region of the country. We want the murder rate to be at most 1. So we want two different things to be true. Here we can use the logical operator *and* which in R is `&`. This operation results in a true only when both logicals are true. To see this consider these examples:

```
TRUE & TRUE
```

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

```
TRUE & FALSE
```

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

```
FALSE & FALSE
```

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

We can form two logicals:

```
west <- murders$region == "West"
safe <- murder_rate <= 1
```

and we can use the `&` to get a vector of logicals that tells us which states satisfy both of our conditions:

```
ind <- safe & west
murders$state[ind]
```

```
## [1] "Hawaii"  "Idaho"   "Oregon"  "Utah"    "Wyoming"
```

`which`

Suppose we want to look up California's murder rate. For this type of operation, it is convenient to convert vectors of logicals into indexes instead of keeping long vectors of logicals. The function `which` tells us which entries of a logical vector are TRUE. So we can type:

```
ind <- which(murders$state == "California")
ind # this is the index that matches the California entry
```

```
## [1] 5
```

```
murder_rate[ind]
```

```
## [1] 3.374138
```

`match`

If instead of just one state we want to find out the murder rates for several, say New York, Florida, and Texas, we can use the function `match`. This function tells us which indexes of a second vector match each of the entries of a first vector:

```
ind <- match(c("New York", "Florida", "Texas"), murders$state)
ind
```

```
## [1] 33 10 44
```

Now we can look at the murder rates:

```
murder_rate[ind]
```

```
## [1] 2.667960 3.398069 3.201360
```

`%in%`

If rather than an index we want a logical that tells us whether or not each element of a first vector is in a second, we can use the function `%in%`. So, say you are not sure if Boston, Dakota and Washington are states, you can find out like this

```
c("Boston", "Dakota", "Washington") %in% murders$state
```

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

Assessments

1. 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.
2. Now use the results from the previous exercise and the function `which` to determine the indices of `murder_rate` associated with values lower than 1.
3. Use the results from the previous exercise to report the names of the states with murder rates lower than 1.
4. Now extend the code from exercises 2 and 3 to report the states in the Northeast with murder rates lower than 1. Hint: Use the previously defined logical vector `low` and the logical operator `&`.

5. 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?

6. Use the match function to identify the states with abbreviations AK, MI, and IA. Hint: Start by defining an index of the entries of `murders$abb` that match the three abbreviations, then use the `[]` operator to extract the states.

7. Use the `%in%` operator to create a logical vector that answers the question: which of the following are actual abbreviations: MA, ME, MI, MO, MU?

8. Extend the code you used in exercise 7 to report the one entry that is **not** an actual abbreviation. Hint: Use the `!` operator, which stands for "not" and turns `FALSE` into `TRUE` and vice-versa, then `which` to obtain an index.