Lecture 3

niceguy

2023-07-31

Vectors Review

As an example, we consider the data of salaries of doctors.

```
offer <- c(241, 590, 533, 425, 261)
offer > 400 # gives vector of booleans
```

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

```
offer[offer > 400] # gives vector of offers greater than 400
```

```
## [1] 590 533 425
```

```
# To understand how it works, consider how
# offer > 400 == [FALSE, TRUE, TRUE, TRUE, FALSE]
# offer[c(FALSE, TRUE, TRUE, TRUE, FALSE)] = [offer[2], offer[3], offer[4]]
```

We can also use not, and, and or.

```
offer[offer > 400 & offer < 550]
```

```
## [1] 533 425
```

```
offer[!(offer > 400 & offer < 550)]
```

```
## [1] 241 590 261
```

```
offer[offer <= 400 | offer >= 550]
```

```
## [1] 241 590 261
```

```
# Essentially De Morgan's Law
```

Vector Operation

Tables

```
spec <- c("family doc", "cardiologist", "ortho", "dermatologist", "psychiatrist")

# We want to know the specialty that earns the most

spec[offer == max(offer)]
```

```
## [1] "cardiologist"
```

```
# max(offer) gives the maximum value
# offer == max(offer) is in the form vector == number, which gives a vector of booleans with at least o
# spec == vector of booleans gives the location of max offer(s)
```

Similarly, to find the position of the maximum salary, one can replace `spec` with the array $(1, 2, 3, \dots)$, so instead of matching the *name* of the specialty, we get the *position* of the specialty.

```
(1:length(spec))[offer == max(offer)]
```

```
## [1] 2
```

Exercise

Write a function with the signature

```
elems_below <- function(vec, upper_bound)
```

The function takes in a vector of numerics vec, and returns a vector that contains the elements of vec that are smaller or equal to `upper_bound`.

```
elems_below <- function(vec, upper_bound){
  vec[vec <= upper_bound]
}

elems_below(c(5, 1, 2, 3), 2)
```

```
## [1] 1 2
```

Write a function that gets the median of a list.

```
my_median <- function(vec){
  vec <- sort(vec)
  len <- length(vec)
  if(len%%2 == 1){
    vec[(len+1)/2]
  }else{
    (vec[len/2] + vec[(len/2) + 1])/2
  }
}
my_median(c(5, 1, 2, 10, 1, 2, 7))
```

```
## [1] 2
```