Vectorization

Most of R’s functions are vectorized, meaning that the function will operate on all elements of a vector without needing to loop through and act on each element one at a time. This makes writing code more concise, easy to read, and less error prone.

x <- 1:4  
x \* 2

## [1] 2 4 6 8

The multiplication happened to each element of the vector.

We can also add two vectors together:

y <- 6:9  
x + y

## [1] 7 9 11 13

Each element of x was added to its corresponding element of y:

x: 1 2 3 4  
 + + + +  
y: 6 7 8 9  
---------------  
 7 9 11 13

Challenge 1

Let’s try this on the mpg column of the mtcars dataset.

Make a new column in the mtcars data frame that is in units of kilometers per liter instead of miles per gallon. Check the head or tail of the data frame to make sure it worked.

Solution to challenge 1

mtcars$kpl <- mtcars$mpg \* 0.43  
head(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb kpl  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4 9.030  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4 9.030  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1 9.804  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1 9.202  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2 8.041  
## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1 7.783

Comparison operators, logical operators, and many functions are also vectorized:

**Comparison operators**

x > 2

## [1] FALSE FALSE TRUE TRUE

**Logical operators**

a <- x > 3 # or, for clarity, a <- (x > 3)  
a

## [1] FALSE FALSE FALSE TRUE

Tip: some useful functions for logical vectors

any() will return TRUE if *any* element of a vector is TRUE.  
all() will return TRUE if *all* elements of a vector are TRUE.

Most functions also operate element-wise on vectors:

**Functions**

x <- 1:4  
log(x)

## [1] 0.0000000 0.6931472 1.0986123 1.3862944

\*\* The lapply function \*\* lapply() takes a function, applies it to each element in a list, and returns the results in the form of a list. If you’ve taken Software Carpentry with R, you may recall that a data frame is actually a special case of a list where all the list items are vectors of the same length.

\*\* What class is each column? \*\*

unlist(lapply(mtcars, class))

## mpg cyl disp hp drat wt qsec   
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"   
## vs am gear carb kpl   
## "numeric" "numeric" "numeric" "numeric" "numeric"

\*\* Divide each column by the mean \*\*

mtcars[] <- lapply(mtcars, function(x) x / mean(x))

#explain

mtcars$mpg/mean(mtcars$mpg)

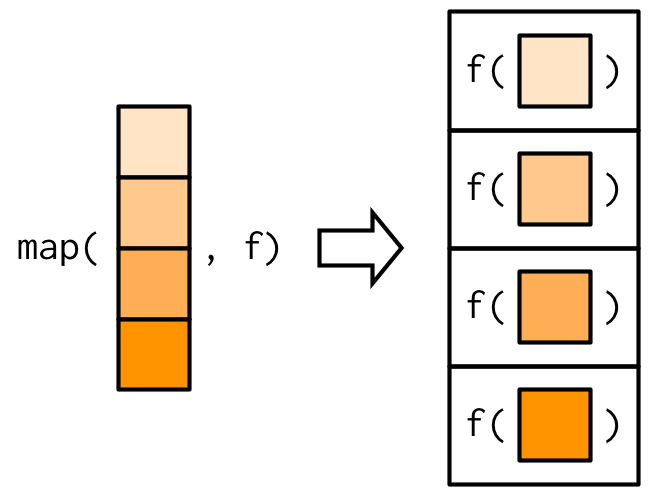
mtcars$mpg/0.5

Challenge 2

Duplicate the code above (to divide each column by the mean) using a for loop.

Solution to challenge 2

for(i in 1:length(mtcars)){  
 print(mtcars[[i]]/mean(mtcars[[i]]))  
}

The functions map() and lapply() work to take input (elements of a list, such as columns of a dataframe, apply a function f() to them, and provide output back [a list, in the case of lapply()]. 

Tip: map() in ‘purrr’ package.

“The tidyverse equivalent to lapply() is map(). The only difference is that lapply() does not support the helpers that you’ll learn about below, so if you’re only using map() from purrr, you can skip the additional dependency and use lapply() directly.” - wickham book