

Introduction to Map Functions

Part 2

Example: A function is defined, which squares a number if it's greater or equal to 0 and cubes it otherwise.

```
square_cube <- function(x){
```

```
  if (x >= 0){
```

```
    y <- x^2
```

```
  } else {
```

```
    y <- x^3
```

```
  }
```

```
  return(y)
```

```
}
```

← It's a boolean expression that should only be TRUE or FALSE

When you attempt to "feed" this function a vector, in this expression, the boolean expression will be:

TRUE FALSE FALSE TRUE

This will cause the codes to crash.

```
numbers <- c(1, -2, -3, 4)
```

```
square_cube(numbers)
```

Output: **Error**

Here is a fix:

```
square_cube_loop <- function(x){  
  for(i in 1:length(x))  
    if (x[i]>= 0){  
      y[i] <- x[i]^2  
    } else {  
      y[i]<- x[i]^3  
    }  
  return(y)  
}
```

```
y<-NA
```

```
square_cube_loop(numbers)
```

Output: 1 -8 -27 16

Adding this for loop fixes the previous coding issue. This will extract the element of the vector one by one.

The first function could work if we use the map function.

vector
↑

→ name of a function

map(numbers, square_cube)

map will apply the function square_cube to every element of the vector "numbers" one by one.

Output:
[[1]]
[1] 1
[[2]]
[1] -8
[[3]]
[1] -27
[[4]]
[1] 16

map gives output as a list. You have to unlist, to make it into a vector.

unlist(map(numbers, square_cube))

Output: 1 -8 -27 16

Data

illustration_data

I selected 6
numeric columns
from the Teams
dataset.

	HBP	SF	RA	ER	ERA	CG
1	NA	NA	303	109	3.55	22
2	NA	NA	241	77	2.76	25
3	NA	NA	341	116	4.11	23
4	NA	NA	243	97	5.17	19
5	NA	NA	313	121	3.72	32
6	NA	NA	266	137	4.95	27
7	NA	NA	287	108	4.30	23
8	NA	NA	362	153	5.51	28
9	NA	NA	303	137	4.37	32

map() Functions

What is the issue when I run this code?

```
median(illustration_data)
```

→ this is a dataframe

Running this code will give you an error because median is a function that needs a "vector" or a specific column from a dataframe.

Map Functions: allow you to apply a function to each element of a list, vector, or dataframe. Map functions allow you to operate on entire vectors or data frames in a single step, without using explicit loops.

dataframe
↑
map(illustration_data, median)
unlist(map(illustration_data, median))

map will apply the function median to every column of the dataframe

Output: HBP SF RA ER ERA CG
NA NA 689 595 3.84 41

Will give you back a named atomic vector.
Meaning every element of the vector actually has a name.

`example_1 <- map_dbl(illustration_data, median)`
`example_1`

→ will give you back a numeric vector or will die trying.
Gives you back a vector instead of a list.

Output:

HBP	SF	RA	ER	ERA	CG
NA	NA	689	595	3.84	41

`example_2 <- map_chr(illustration_data, median)`
`example_2`

→ will give you back a character vector or will die trying.

Output:

HBP	SF	RA	ER	ERA	CG
NA	NA	"689"	"595"	"3.84"	"41"

↗ gives you back a dataframe

```
example_3 <- map_dfr(illustration_data, median)
```

```
example_3
```

Output:

HBP
NA

SF
NA

RA
689

ER
595

ERA
3.84

CG
41

} Is a
dataframe,
not a
vector.

Map Functions when you have to use arguments

dataframe
↑

```
map_dbl(illustration_data, median, na.rm = TRUE)
```

argument of median

Output:

IBP	SF	RA	ER	ERA	CG
44	44	689	595	3.84	41

The 1st argument of median, which is a vector will be accounted for by every column of illustration-data.

Another notation: This is a different notation, but will give you some result as above

```
map_dbl(illustration_data, \(x) median(x, na.rm = TRUE))
```

Output:

IBP	SF	RA	ER	ERA	CG
44	44	689	595	3.84	41

More Examples - Map Function - Iterating Over a Single Vector

illustration_data_2

Got this from
the Teams
dataframe

	teamID	name	began	ended
1	LAA	Los Angeles Angels	1961	1964
2	CAL	California Angels	1965	1996
3	ANA	Anaheim Angels	1997	2004
4	LAA	Los Angeles Angels of Anaheim	2005	2022

Example: We want to find out the number of characters in the team name.

`map_dbl(illustration_data_2, str_length)`

dataframe → `illustration_data_2`
string length → `str_length`

Output: Error (because it has to give back a vector)

`map_dbl(illustration_data_2$name, str_length)`

vector → `illustration_data_2$name`

Output: 18 17 14 29

`map_dfr(illustration_data_2, str_length)`

output is a dataframe → `map_dfr`
dataframe → `str_length`

Output:

TeamID	name	began	ended
3	18	4	4
3	17	4	4
3	14	4	4
3	29	4	4

output is a dataframe

More Examples - Map Function - Iterating Over Our Own Function

```
top5 <- function(team_name, data) {  
  output <- data %>%  
    filter(name == team_name) %>%  
    select(teamID, yearID, W, L, name) %>%  
    arrange(desc(W)) %>%  
    head(n = 5)  
  return(output)  
}
```

This is a function that
needs the dataframe
name and a team name
as an argument.

`top5(team_name = "New York Yankees", data = Teams)`

name of
the team

name of the
dataframe

Output:

	teamID	yearID	W	L	name
1	NYA	1998	114	48	New York Yankees
2	NYA	1927	110	44	New York Yankees
3	NYA	1961	109	53	New York Yankees
4	NYA	1932	107	47	New York Yankees
5	NYA	1939	106	45	New York Yankees

```
map_dfr(illustration_data_2$name, top5, Teams)
```

↓
output will
be a dataframe

This is a vector
that came from
illustration_data

will apply this function
↑ to every element
of the vector

data =

↓ the 2nd
argument
for the function
top5

illustration_data_2

	teamID	name	began	ended
1	LAA	Los Angeles Angels	1961	1964
2	CAL	California Angels	1965	1996
3	ANA	Anaheim Angels	1997	2004
4	LAA	Los Angeles Angels of Anaheim	2005	2022

`map_dfr(illustration_data_2$name, top5, Teams)` ^{data =}

Output:

output
is a
dataframe.

	teamID	yearID	W	L	name
1	LAA	1962	86	76	Los Angeles Angels
2	LAA	1964	82	80	Los Angeles Angels
3	LAA	1961	70	91	Los Angeles Angels
4	LAA	1963	70	91	Los Angeles Angels
5	CAL	1982	93	69	California Angels
6	CAL	1986	92	70	California Angels
7	CAL	1989	91	71	California Angels
8	CAL	1985	90	72	California Angels
9	CAL	1979	88	74	California Angels
10	ANA	2002	99	63	Anaheim Angels
11	ANA	2004	92	70	Anaheim Angels
12	ANA	1998	85	77	Anaheim Angels
13	ANA	1997	84	78	Anaheim Angels
14	ANA	2000	82	80	Anaheim Angels
15	LAA	2008	100	62	Los Angeles Angels of Anaheim
16	LAA	2014	98	64	Los Angeles Angels of Anaheim
17	LAA	2009	97	65	Los Angeles Angels of Anaheim
18	LAA	2005	95	67	Los Angeles Angels of Anaheim
19	LAA	2007	94	68	Los Angeles Angels of Anaheim

} applied the function on Los Angeles Angels
} applied the function to California Angels
} applied the function to Anaheim Angels
} applied the function to Los Angeles Angels of Anaheim