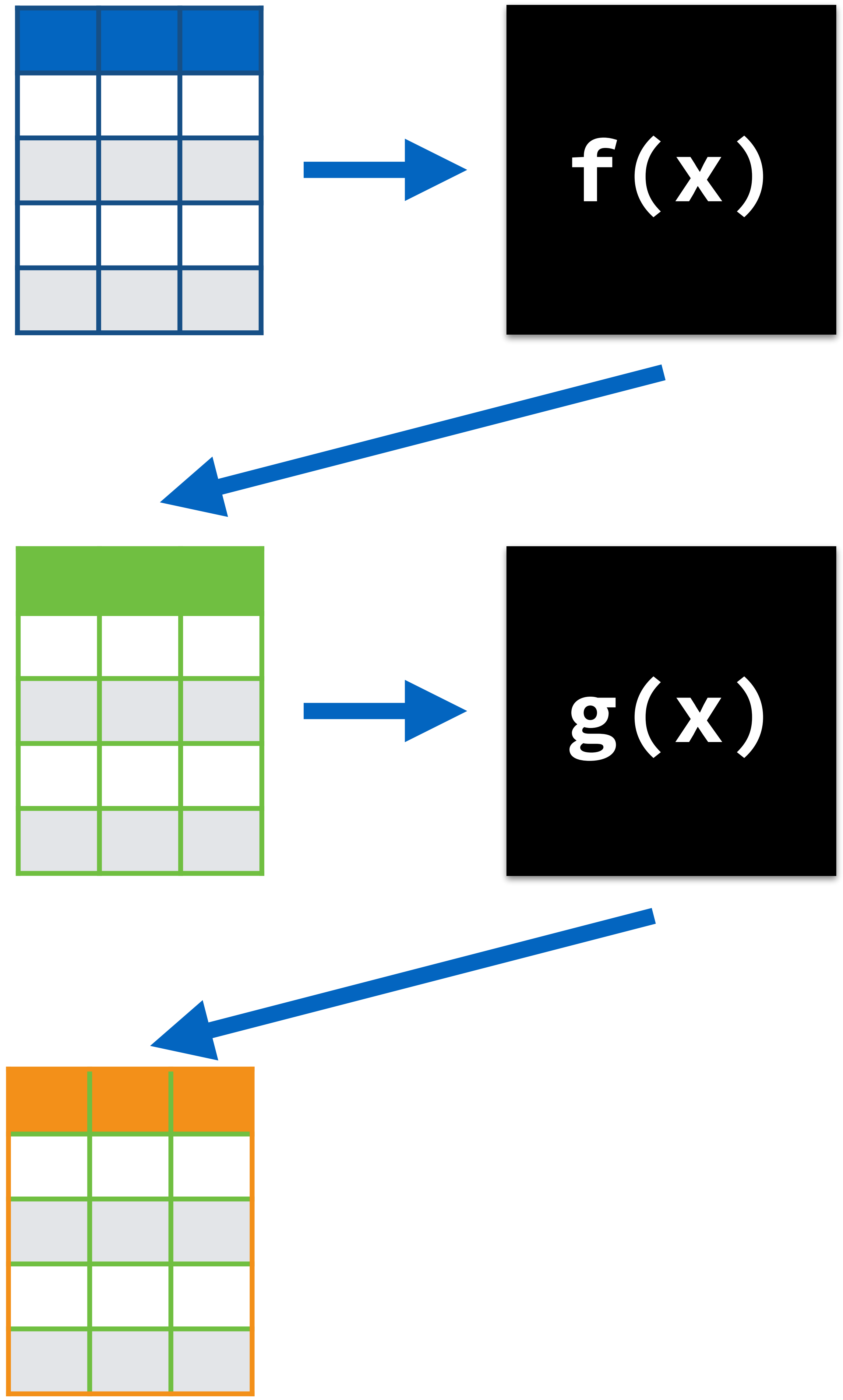
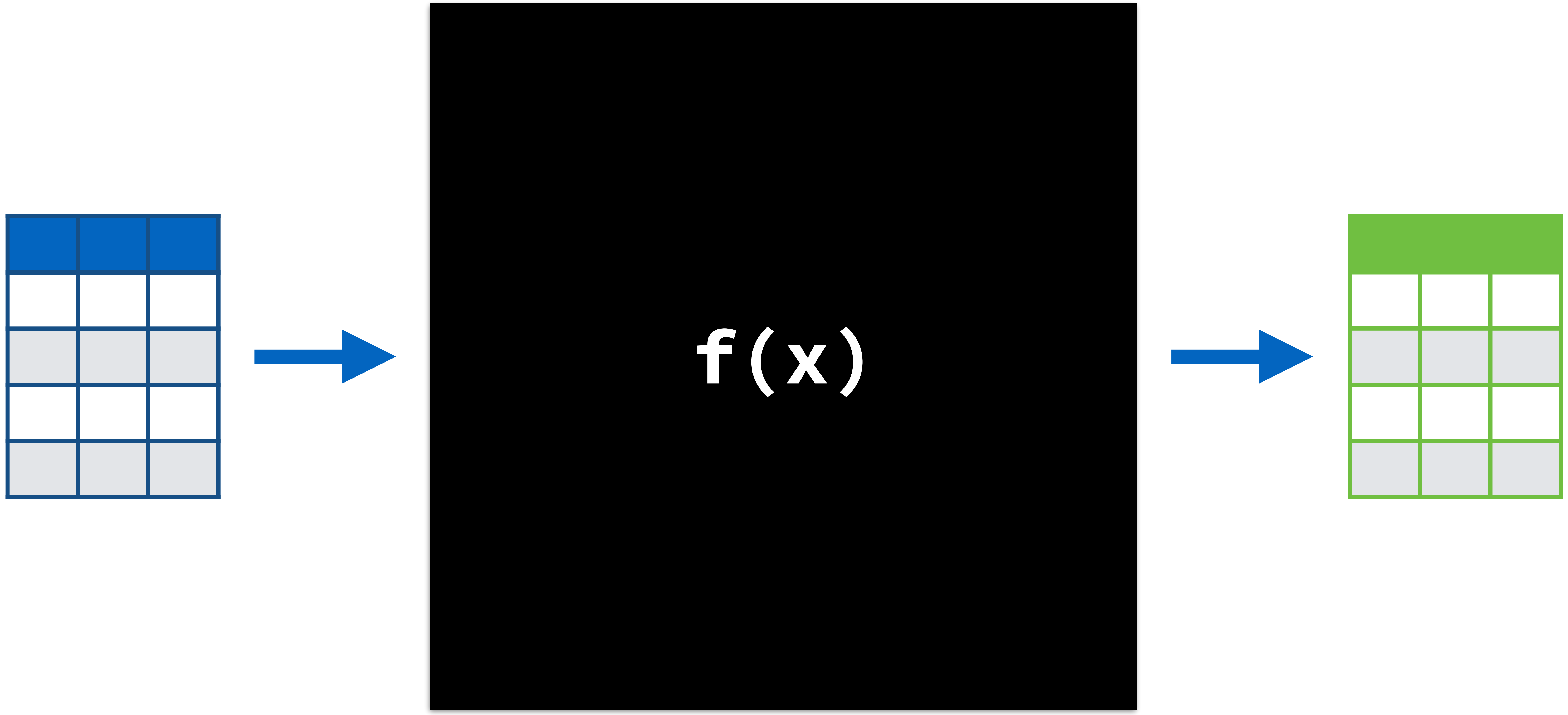
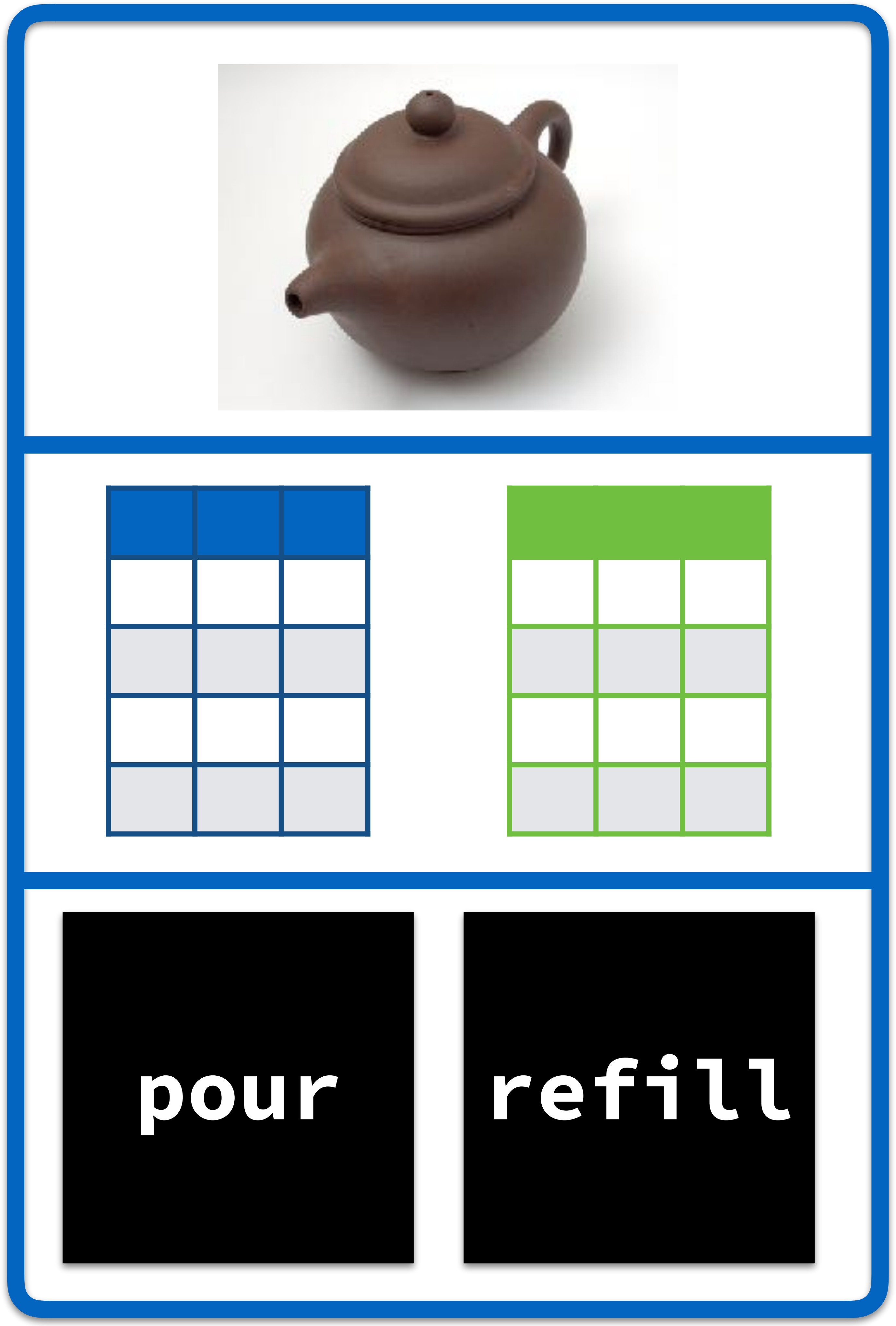
OBJECT-ORIENTED PROGRAMMING IN R: S3 & R6

What is Object-Oriented Programming?


f(x)
g(x)

```
calculate_something <- function(x, y, z) {
  # do something
  return(the_result)
}
```





pour

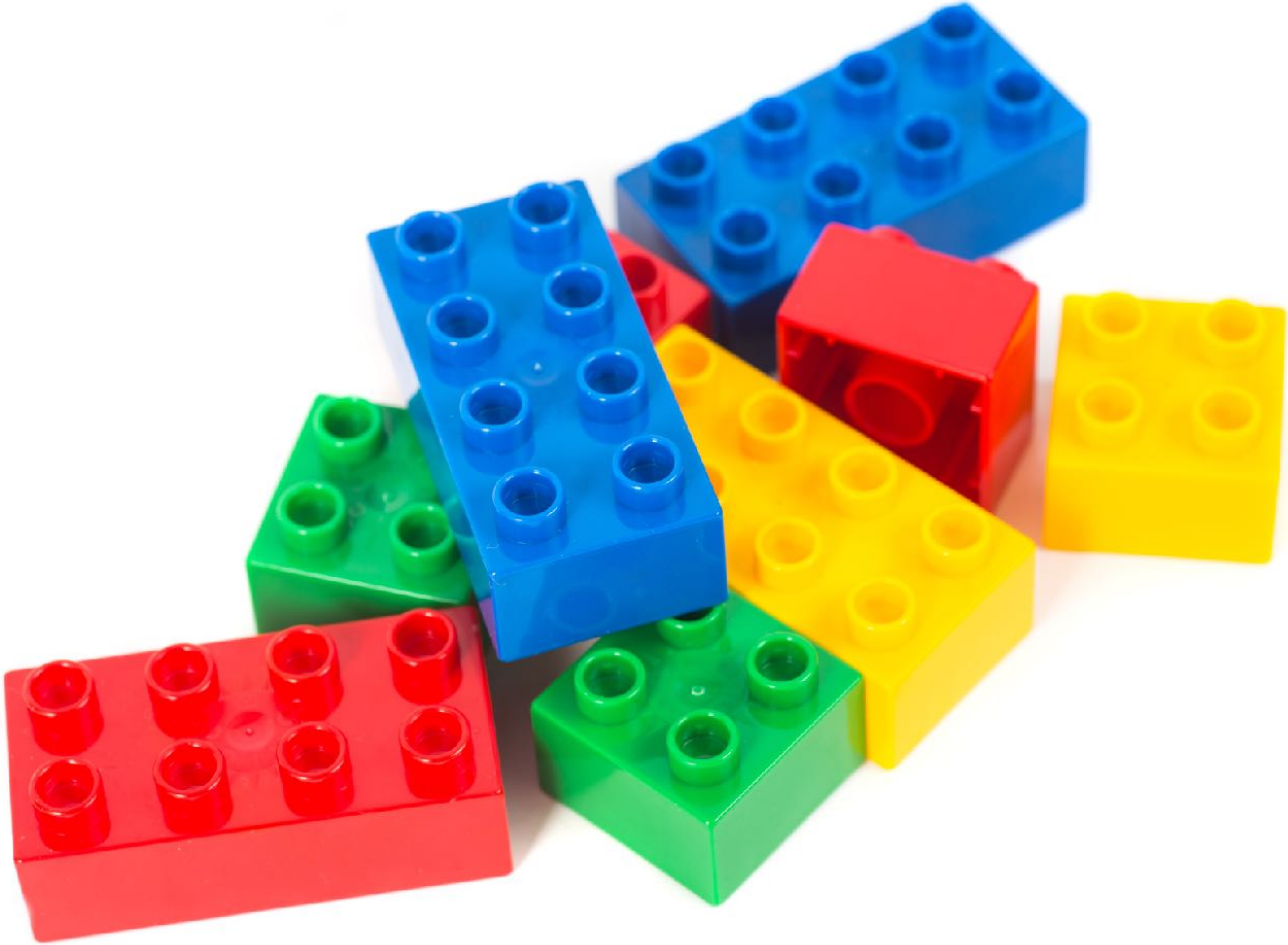
refill

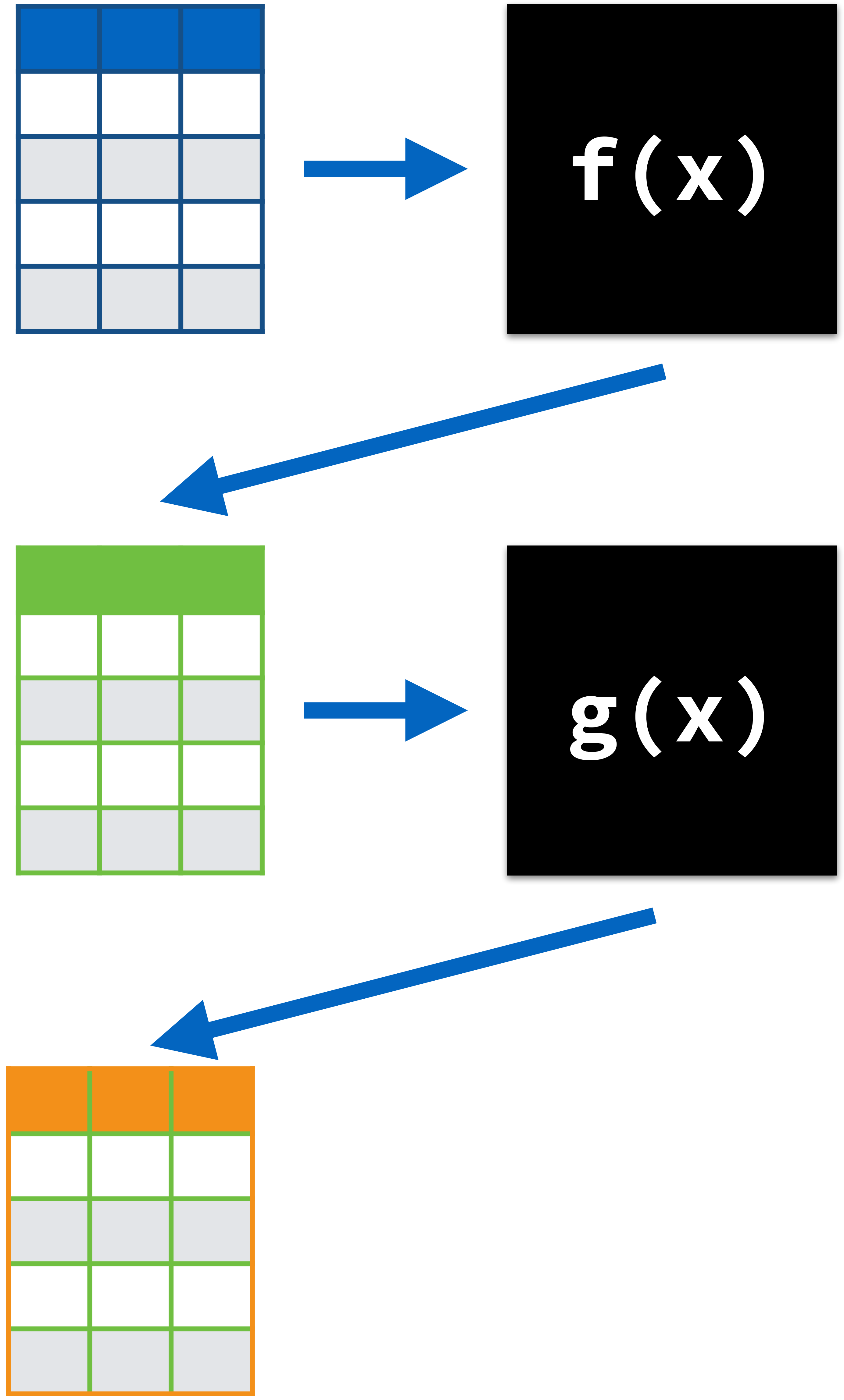
A **method** is just a function, talked about in an OOP context

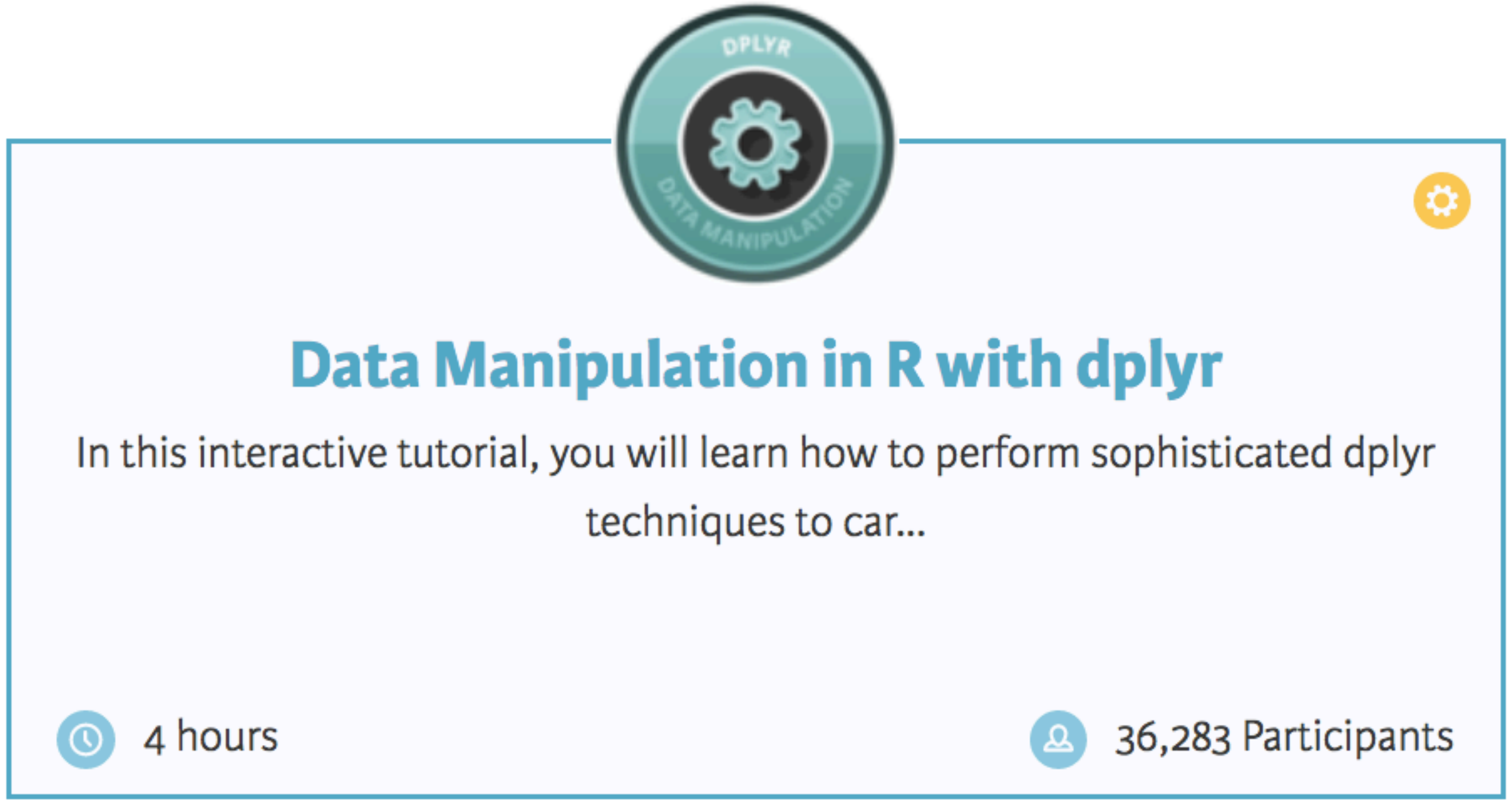
logical vector	closure function
integer vector	builtin function
numeric vector	special function
complex vector	environment
character vector	null
raw vector	formula
list	expression
matrix	call
array	pairlist
data.frame	external pointer
factor	

`list`

`environment`




f(x)
g(x)



Data Manipulation in R with dplyr

In this interactive tutorial, you will learn how to perform sophisticated dplyr techniques to car...

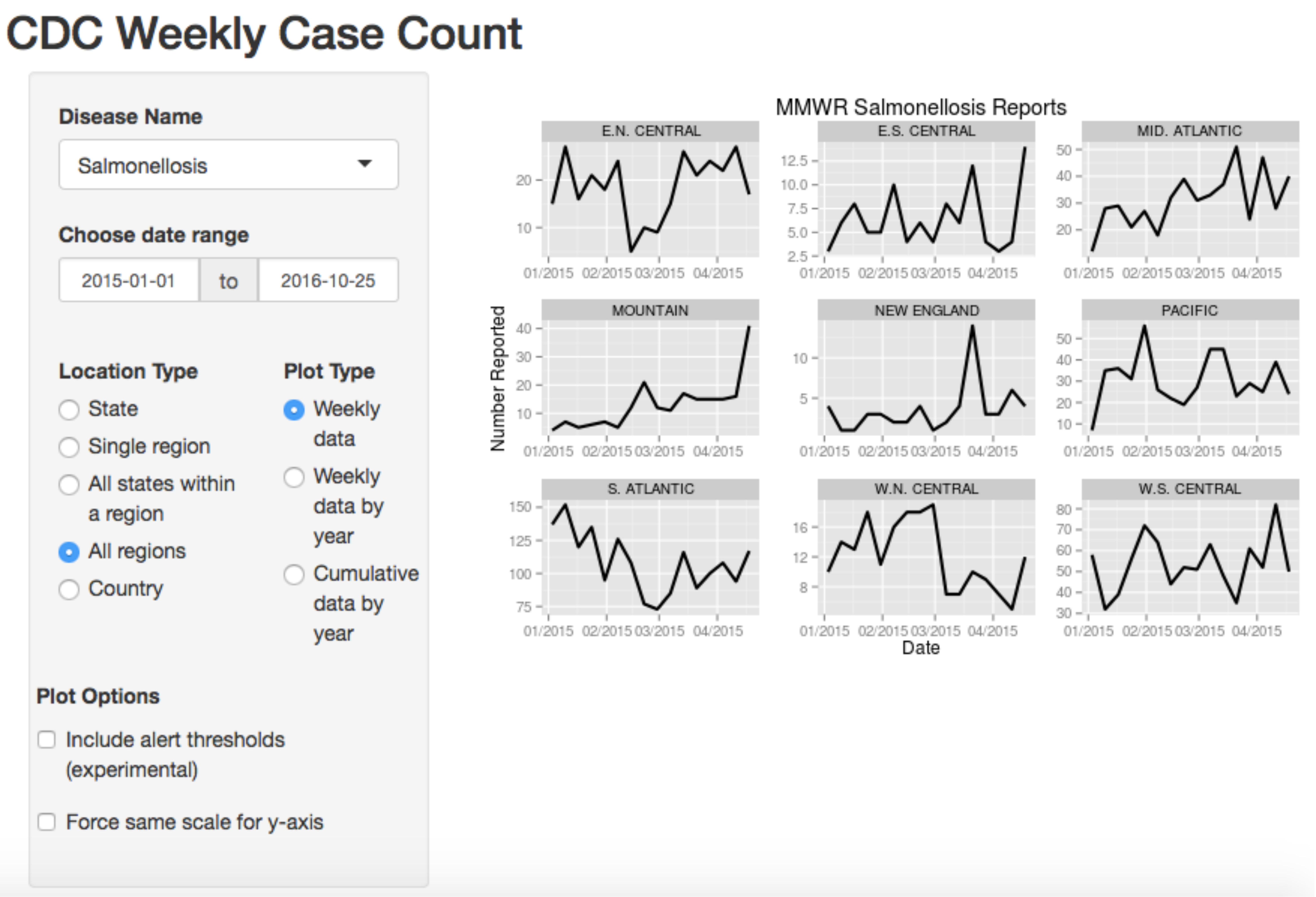
4 hours

36,283 Participants

When is OOP a good idea?

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

WIKIPEDIA



CDC Weekly Case Count

Disease Name

Salmonellosis ▾

Choose date range

2015-01-01 to 2016-10-25

Location Type

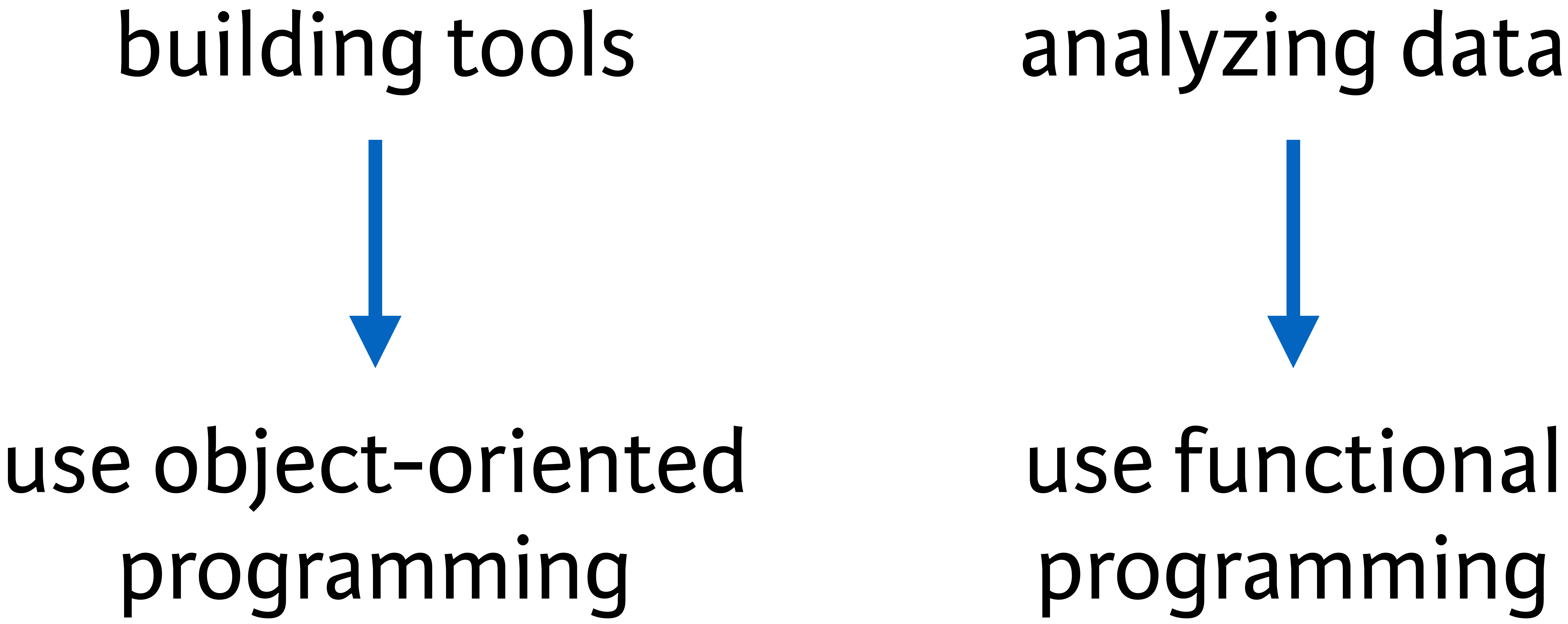
- ○ State
- ○ Single region
- ○ All states within a region
- ● All regions
- ○ Country

Plot Type

- ● Weekly data
- ○ Weekly data by year
- ○ Cumulative data by year

Plot Options

- □ Include alert thresholds (experimental)
- □ Force same scale for y-axis



building tools

use object-oriented programming

analyzing data

use functional programming

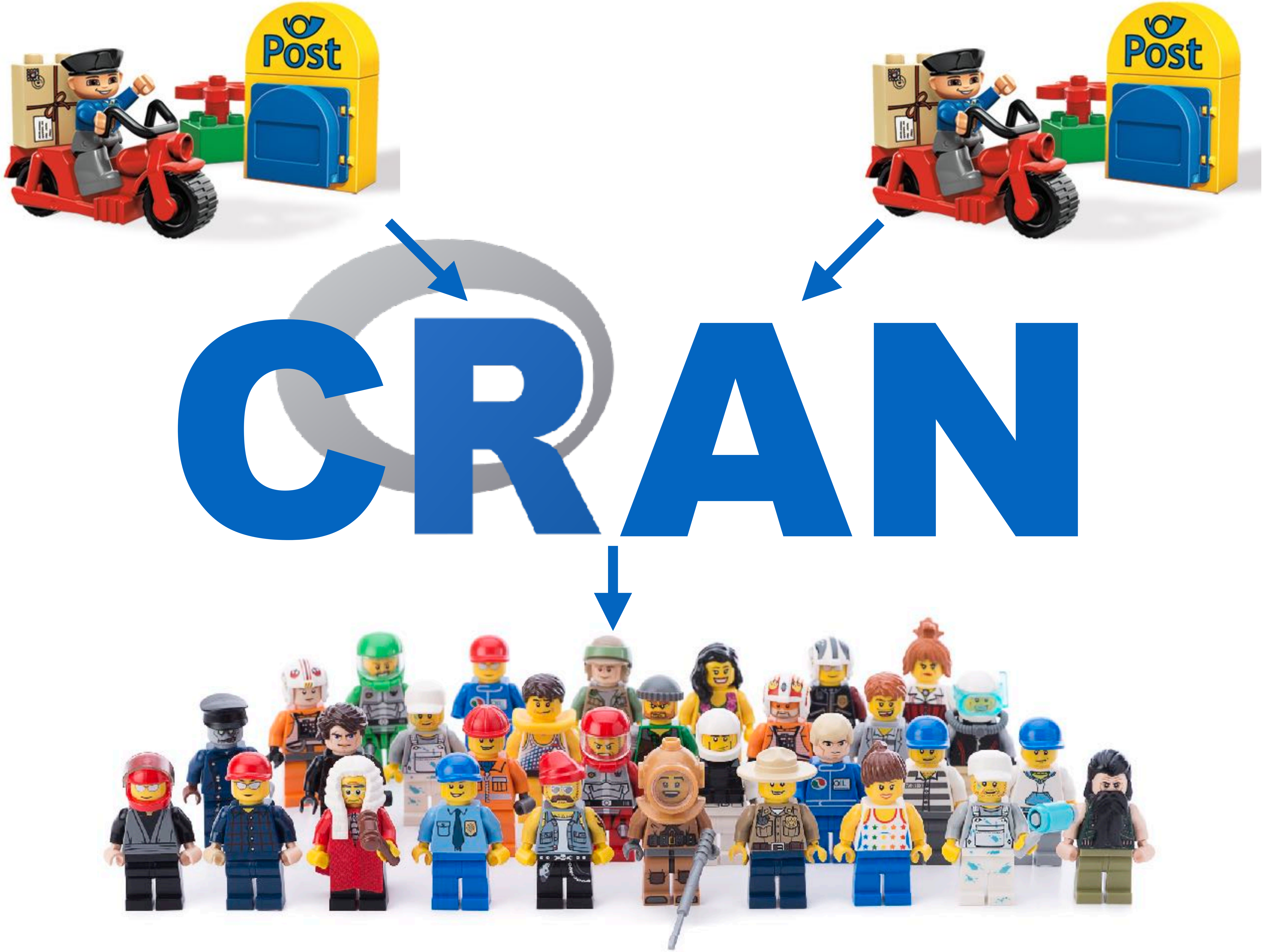
Summary

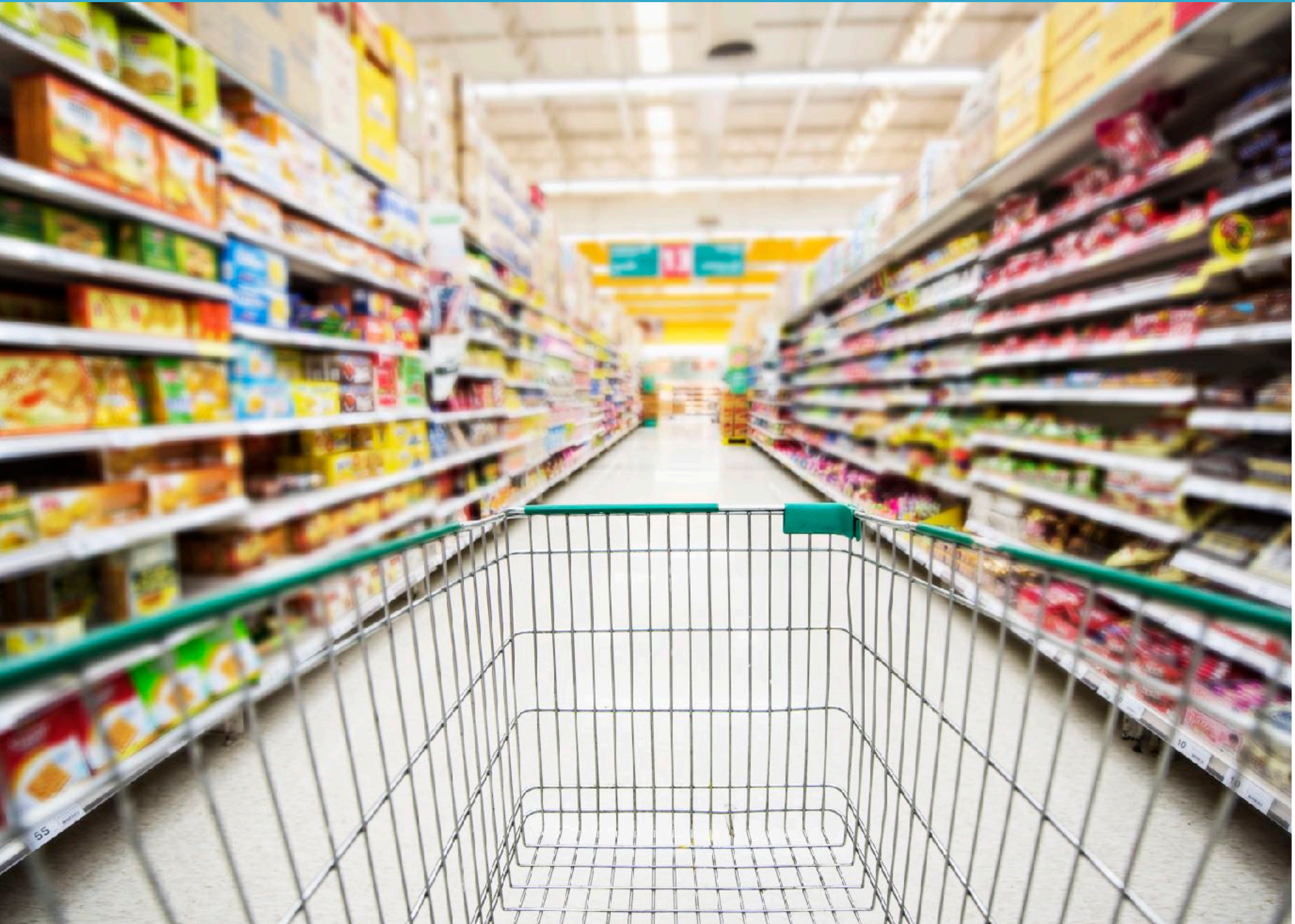
- With **functional programming**, think about the **functions first**.
- With **object-oriented programming** (OOP) think about the **data structures first**.
- **Don't** use OOP for **general purpose data analyses**.
- **Do** use OOP when you have **a limited number of complex objects**.

OBJECT-ORIENTED PROGRAMMING IN R: S3 & R6

Let’s practice!

The Nine Systems


Post
Post
CRAN



ReferenceClasses

R.oo

S4

OOP

R5

S3

R6

proto

mutatr

Summary

- Use **S3 regularly**
- Use **R6** when you need **more power**
- Use **S4** for **Bioconductor**
- **Maybe** use **ReferenceClasses**

Let's practice!

How does R Distinguish Variables?

```
> str(sleep)
'data.frame':20 obs. of  3 variables:
 $ extra: num  0.7 -1.6 -0.2 -1.2 -0.1 ...
 $ group: Factor w/ 2 levels "1","2": 1 1 1 1 1 ...
 $ ID   : Factor w/ 10 levels "1","2","3","4",..: 1 2..
```

```
> class(sleep)
[1] "data.frame"
```

```
> (int_mat <- matrix(1:12, 3))
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

```
> class(int_mat)
[1] "matrix"
```

```
> typeof(int_mat)
[1] "integer"
```

```
> (num_mat <- matrix(rnorm(12), 3))
           [,1]       [,2]        [,3]       [,4]
[1,] -0.2911535 -0.1139933 -0.71290868  0.8640191
[2,] -2.2266419 -1.3604316 -1.90716974  0.4012884
[3,] -0.7504663 -1.2478873  0.01104117 -0.8127333
```

```
> class(num_mat)
[1] "matrix"
```

```
> typeof(num_mat)
[1] "double"
```

```
mode()
storage.mode()
```

Summary

- `class()` is your **first choice** for determining the kind of variable

- `typeof()` is also **occasionally useful**

- `mode()` and `storage.mode()` are old functions; **don't use them**

OBJECT-ORIENTED PROGRAMMING IN R: S3 & R6

Let’s practice!

OBJECT-ORIENTED PROGRAMMING IN R: S3 & R6

Assigning Classes

```
> (x <- rexp(10))
 [1] 0.195051 2.191040 0.498703 0.976122 0.299001
 [6] 0.105187 0.090073 2.328233 3.043201 2.129631
```

```
> class(x) <- "random_numbers"
```

```
> x
 [1] 0.195051 2.191040 0.498703 0.976122 0.299001
 [6] 0.105187 0.090073 2.328233 3.043201 2.129631
attr(,"class")
[1] "random_numbers"
```

```
> class(x)
[1] "random_numbers"
```

```
> typeof(x)
[1] "double"
```

```
> is.numeric(x)
[1] TRUE
```

```
> length(x)
[1] 10
```

```
> mean(x)
[1] 1.1856
```

Summary

- You can **override** the **`class()`**
- This **won't** break existing functionality

OBJECT-ORIENTED PROGRAMMING IN R: S3 & R6

Let's practice!