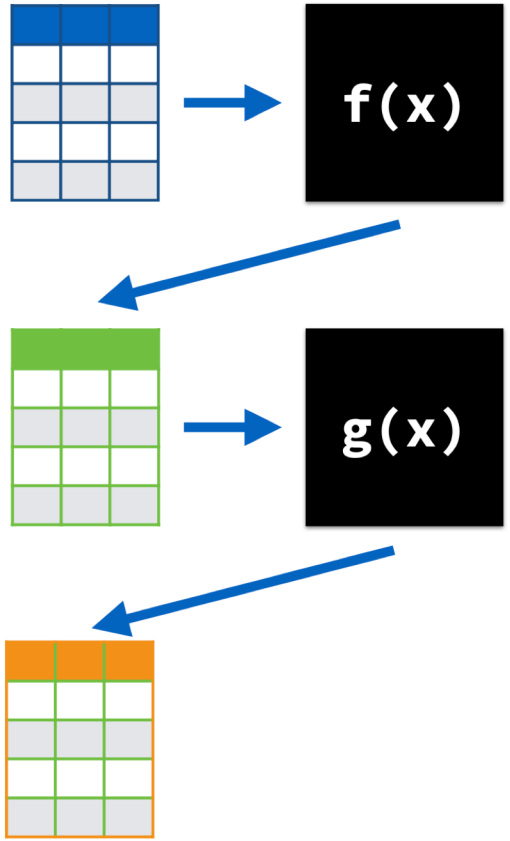
# Object-Oriented Programming in R

Turgut Yigit Akyol

23.03.2021

# Introduction

- Generally in R, **functional programming** is much more important than object-oriented programming.




building tools → use object-oriented programming

analyzing data → use functional programming

# Introduction

Reference Classes

S3

R.oo

OOP

S4

R6

R5

mutatr

proto

S3 → 


S4 → 


Reference Classes → 

# Introduction

- You first need to define a **constructor** function which creates new instances of your new class.
- Arguments to this constructor function should match the items that forms the object from the new class.
- You also define the structure of the object from this new class in the constructor function.
- You then define **methods**, which are the functions associated with the new class.

# S3

```
# Method to create an instance -------------------
Student = function(name, grad_year, credits, id, courses) {
  output = list('name' = name, 'grad_year' = grad_year, 'credits' = credits,
                'id' = id, 'courses' = courses)
  class(output) = 'Student'
  return(output)
}

# Method for greeting ----------------------------
hello = function(student) {
  UseMethod('hello')
}

hello.Student = function(student) {
  paste('Hi! My name is', student$name)
}
```

# S3

## Usage of the constructor function

```
18 # Create instance -------------------------------
19 turgut = Student(name = 'Turgut',
20                  grad_year = 2018,
21                  credits = 42,
22                  id = 'au656527',
23                  courses = list('OOP in R')
24 )
```

New object

Constructor

Arguments
(elements of the object)

```
> hello(turgut)
[1] "Hi! My name is Turgut"
>
> turgut$id
[1] "au656527"
>
> turgut$grad_year
[1] 2018
>
```

Access the elements of the new object with **$** sign.

# S4

```
# Initiate the class ---------------------------
setClass('Student',
         slots = list(
           name = 'character',
           grad_year = 'numeric',
           credits = 'numeric',
           id = 'character',
           courses = 'list')
         )

# Method to create an instance -----------------
setGeneric('Student', function(name, grad_year, credits, id, courses) {
  standardGeneric('Student')}
  )

setMethod('Student',
          c(name = 'character',
            grad_year = 'numeric',
            credits = 'numeric',
            id = 'character',
            courses = 'list'),
          function(name, grad_year, credits, id, courses) {
            output = new(Class = 'Student',
                         name = name, grad_year = grad_year, credits = credits,
                         id = id, courses = courses)
            return(output)
          }
)

# Method for greeting
setGeneric('hello', function(student) {
  standardGeneric('hello')}
)

setMethod('hello',
          c(student = 'Student'),
          function(student) {
            paste('Hi! My name is', student@name)
          }
)
```

# S4

## Usage of the constructor function

```
18 # Create instance -------------------------------
19 turgut = Student(name = 'Turgut',
20                  grad_year = 2018,
21                  credits = 42,
22                  id = 'au656527',
23                  courses = list('OOP in R')
24 )
```

New object

Constructor

Arguments
(elements of the object)

```
> hello(turgut)
[1] "Hi! My name is Turgut"
>
> turgut@id
[1] "au656527"
>
> turgut@grad_year
[1] 2018
>
```

Access the elements (**slots**) of the new object with **@** sign.

# Reference Classes

```
Student = function(name, grad_year, credits, id, courses) {
  output = setRefClass('Student',
                       fields = list(
                         name = 'character',
                         grad_year = 'numeric',
                         credits = 'numeric',
                         id = 'character',
                         courses = 'list'),
                       methods = list(
                         hello = function() {
                           paste('Hi! My name is', name)
                         },
                         add_credits = function(n) {
                           credits <<- credits + n
                         },
                         get_email = function() {
                           paste0(id, '@mbg.au.dk')
                         })
                       )
   return(output$new(name = name,
                     grad_year = grad_year,
                     credits = credits,
                     id = id,
                     courses = courses))
}
```

Complex assignment operator (<<–) is used to modify one of the fields of an object with a method.

# Reference Classes

## Usage of the constructor function

```
18 # Create instance --------------------------------
19 turgut = Student(name = 'Turgut',
20                  grad_year = 2018,
21                  credits = 42,
22                  id = 'au656527',
23                  courses = list('OOP in R')
24 )
```

New object

Constructor

Arguments
(elements of the object)

```
> turgut$hello()
[1] "Hi! My name is Turgut"
>
> turgut$get_email()
[1] "au656527@mbg.au.dk"
>
> turgut$credits
[1] 42
>
> turgut$add_credits(4)
>
> turgut$credits
[1] 46
>
```

Objects are **mutable**, i.e. they can be modified in place.

Access the elements (**fields**) and **methods** of the new object with **$** sign.

# Further Reading

- **Advanced R** by Hadley Wickham
  https://adv-r.hadley.nz/

- **R Packages** by Hadley Wickham and Jenny Bryan
  https://r-pkgs.org/