

OOP and S3 types in R

Ravichandran S.

ABCS, BIDS,
FNLCR

Scope

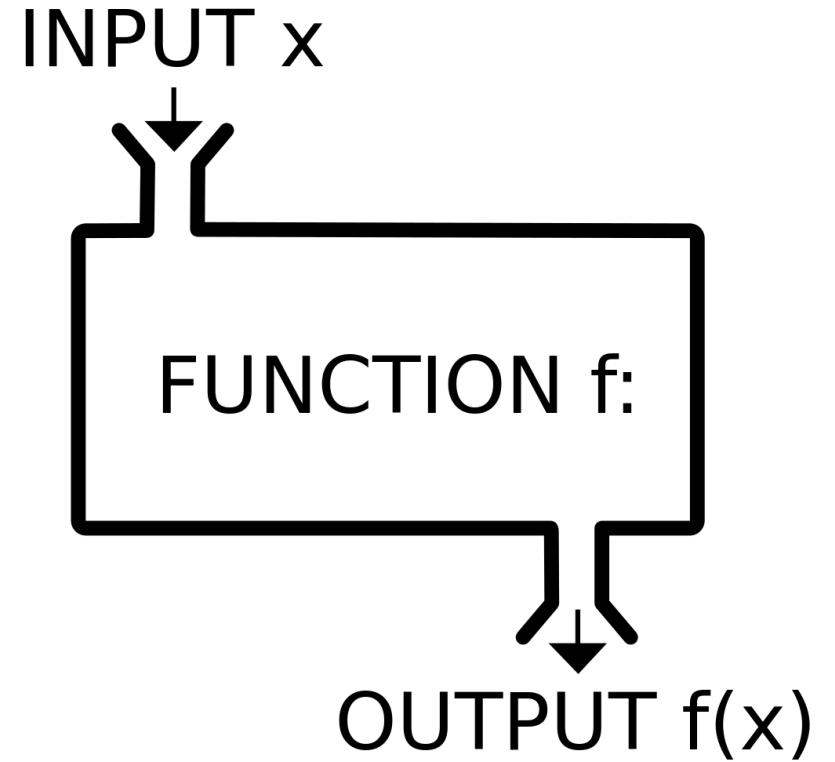
- OOP
 - Concepts might be similar to other languages, but ...
- Specific to R
- Examples

Specific goals

- Note I am !here to teach OOP
- Reinforce concepts that you already know; associate them with OOP. In that process, I will remind/provide some definitions/examples of OOP
- Specific to R ; Easy for other programmers

Functional programming

- Commonly used
- Focus is on functions
- Chain functions together to accomplish things
- Good for?
 - Data analysis, modeling etc.

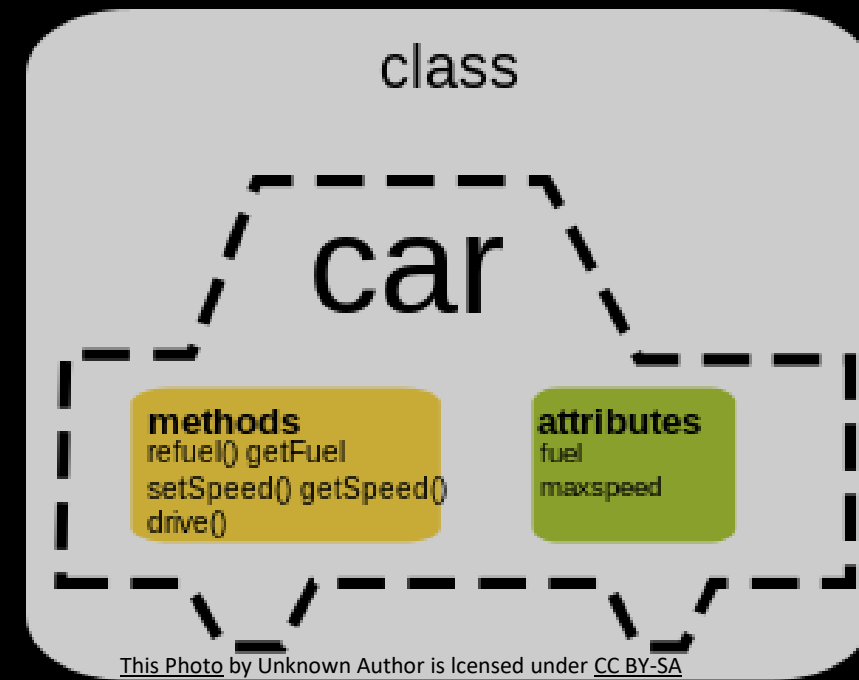


[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

```
my_add <- function(x,y)
{
  # do some task
  return(x + y)
}
```

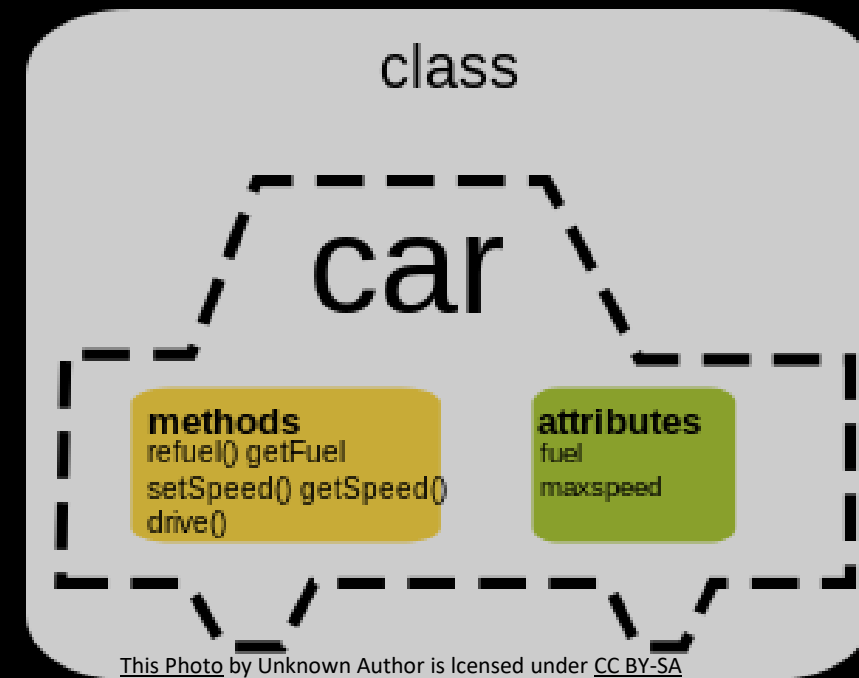
OOP

- Focus on objects
- Defines Object
- Describe its attributes (size, seats etc.)
- Define methods to describe what object can do
 - Note in OOP, functions are called METHODS



OOP

- What is OOP good for?
 - Developing tools, GUIs
 - Complex limited # of objects
 - Specifically when you know you can define the objects clearly
 - Developing GUIs (limited # of options)
 - Interface that can handle limited # of inputs ex.
 - Bioconductor objects (complex but can be reused)



Object types in R

- ~ 20 types
- Integer, logical, numeric, data.frame, List, matrix, array, factor, formula, environment, etc
- Most important types (create complex objects are:
 - List
 - Environment
- These constitute the building blocks that are needed for analysis

Interrogating the variables

- For OOP to work, R has to identify the class of the variables
- How does R identify the class variables?
- Class (command: *class*)
 - Doesn't tell the whole story
- Typeof (command: *typeof* ; c-code)
 - Supplements *class* command

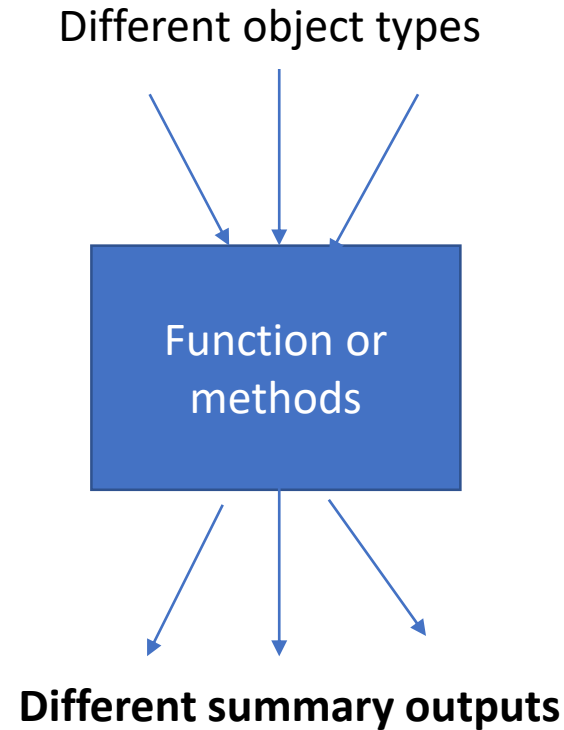


Hands-on 1

Object Types
Class of objects

A simple function to show that it behaves differently for different input types

OOP concept: Functions behave differently for different objects



```
>
>
> x_num <- rnorm(50)
> x_fac <- factor(sample(letters[1:10],50,replace=T))
> model <- lm(mpg ~ wt, mtcars)
>
>
> summary(x_num)
      Min.   1st Qu.   Median     Mean   3rd Qu.    Max.
-2.50817 -0.64324 -0.08345 -0.14963  0.47288  1.71476
> summary(x_fac)
  a  b  c  d  e  f  g  h  i  j
6  1  5  9  3  6  7  1 10  2
> summary(model)

Call:
lm(formula = mpg ~ wt, data = mtcars)

Residuals:
      Min       1Q   Median       3Q      Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776   19.858 < 2e-16 ***
wt          -5.3445     0.5591   -9.559 1.29e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.046 on 30 degrees of freedom
Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

Polymorphism; Function Overloading

OOP Systems (frameworks) in R

- Important systems
- S3 (Introduced in 3rd version of S Language)
- S4 (4th version of S)
 - Bioconductor
- R6 (introduced in 6 version of S; more matured)
- ReferenceClasses (RC)

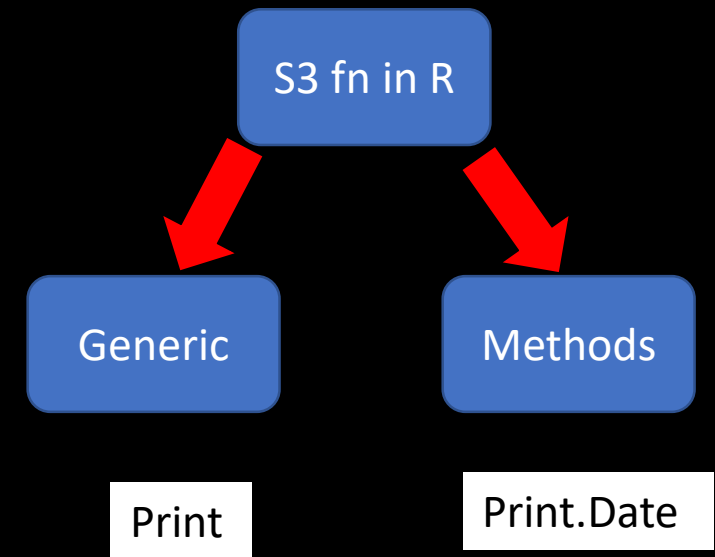
***You can think
of the systems
as different
packages for
implementing
OOP***

S3 System

- Why learn S3?
 - Most commonly used type
 - Simple; lacks formal definition
 - Freedom to be creative (comes with cost!)
 - Create custom class of objects and use S3 to accomplish complex tasks

S3 Object System in R

- Central players
 - Class & Method
- CLASS
 - defines type of object, its properties, how it works with other objects
- METHOD
 - Function associated with a particular object type
- R OO style is different than C++ or Java etc
- In S3 a generic function will decide what appropriate method to call



Generics and Methods

- If we have no overloading then we need a lot more functions
- S3 was created to solve this problem
 - Takes a function for each class and splits into two parts:
 - generic function & method function

How to name a Method?

- Standard notation for S3
- generic.class
- Arguments should be same for both generic and UseMethod
- To avoid from being mistaken, don't name your variable/function with "dot"
 - DON'T: my.print
 - Maybe: my_print_function

```
> print
function (x, ...)
  UseMethod("print")
<bytecode: 0x000000001e689540>
<environment: namespace:base>
```

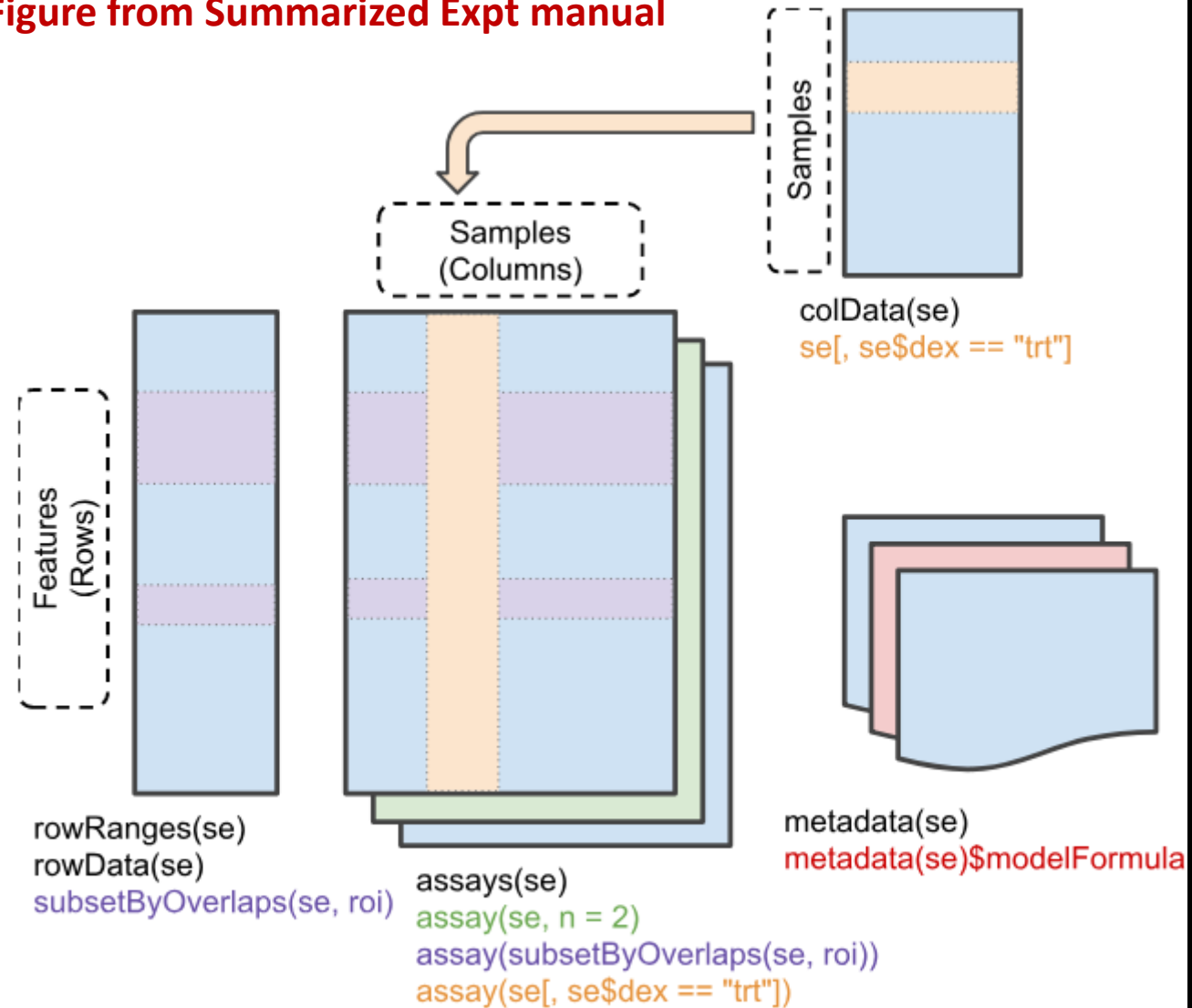
	<i>UseMethod</i>
<i>Generic</i>	generic.class
print	print.data.frame
	print.data.table*
	print.Date
	print.default
	print.dendrogram

Hands-on 2

S4

- Very useful to create new class
 - Ex SummarizedExperiment
- Complex objects
 - Genomic objects
 - Elements of class are called slots
 - SetMethod to define methods for a class
- Reused in many contexts

Figure from Summarized Expt manual



Acknowledgements

- Statistics for lunch team
- Presentations/Lectures/Tutorials from the following people
- Hadley Wickham
 - [Adv. R tutorial](#)
- Richie Cotton
 - [Youtube](#)
- Kelly Black, Univ Georgia
 - [cyclismo.org](#)

THANK YOU