



Paradigmas de Programación en R

VERÓNICA JIMÉNEZ JACINTO

UNIDAD UNIVERSITARIA DE SECUENCIACIÓN MASIVA Y BIOINFORMÁTICA

INSTITUTO DE BIOTECNOLOGÍA

UNAM

Objetivos

Hacer una breve introducción a los paradigmas de programación mas populares para analizar cuales de estos usa el lenguaje de Programación R, que lo hace tan versátil y popular.

INDICE

- ▶ ¿Qué es un paradigma de programación?
- ▶ Paradigmas de programación
 - ▶ Programación Estructurada
 - ▶ Programación lógica
 - ▶ Programación funcional
 - ▶ Programación Orientada a Objetos
- ▶ ¿Que es R?
- ▶ Ambientes de trabajo de R

¿Qué es un paradigma?



¿Qué lenguajes de programación conoces?



¿Qué es un Paradigma?

Un modelo de trabajo compartido por una comunidad científica, cuyos miembros están de acuerdo en qué es un problema legítimo y cuál es una solución legítima del problema. La adopción de un paradigma hace posible compartir conceptos básicos, procedimientos, etc. Por otra parte, las revoluciones científicas han emergido de cambios en los paradigmas dominantes.

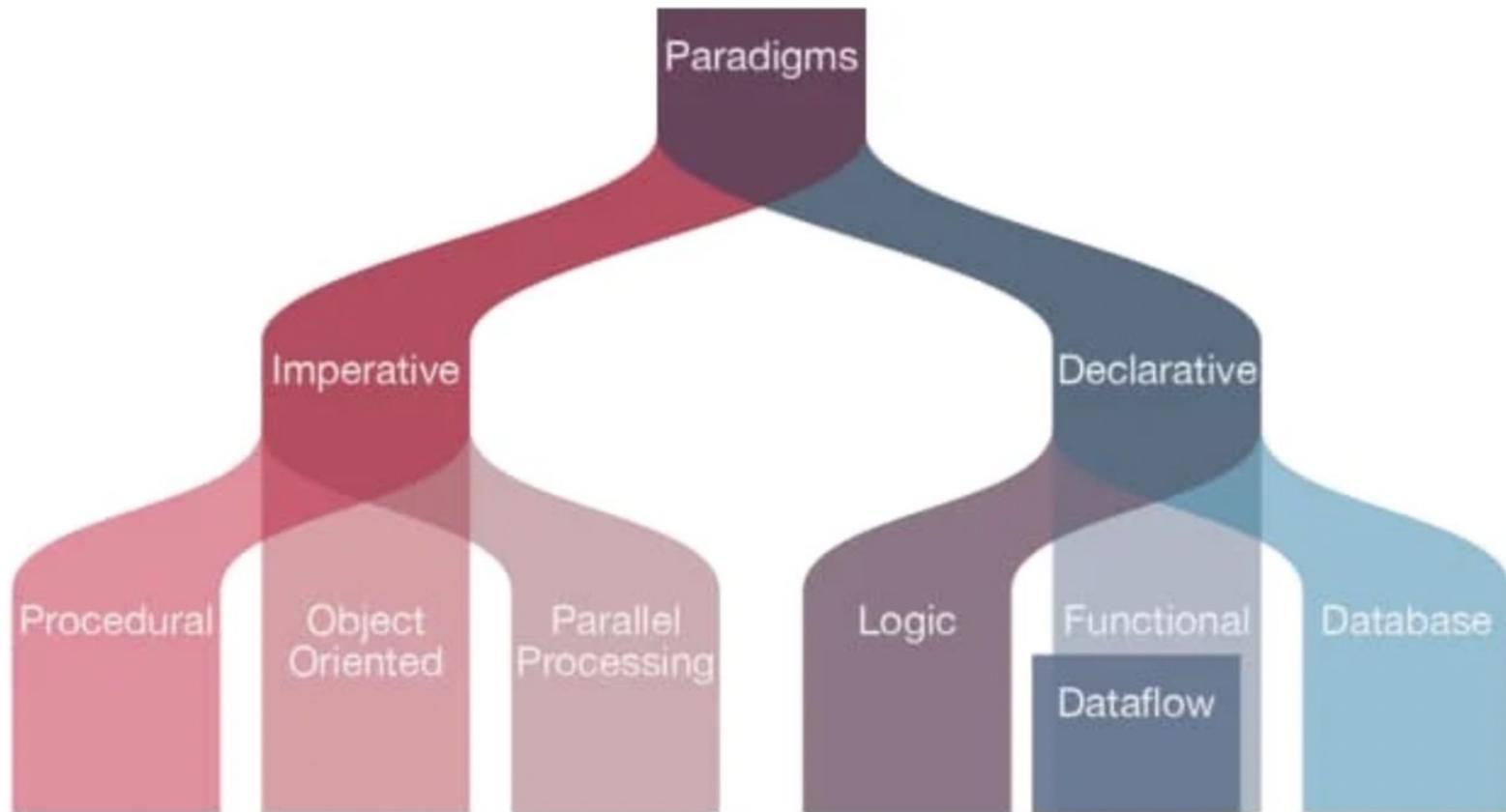
¿Por qué hablar de paradigmas de programación?

El estudio de los paradigmas constituye la fuente principal en la formación del estudiante, para preparar su integración a una comunidad científica particular donde llevará a cabo su práctica. En las Ciencias de la Computación es posible observar a diversas comunidades, hablando sus propios lenguajes y usando sus propios paradigmas. De hecho, los lenguajes de programación estimulan el uso de algunos paradigmas, mientras que inhiben notoriamente otros.

¿Por qué hablar de paradigmas de programación?

Las dificultades típicas presentes en el desarrollo de programas de cómputo son reflejo de un **repertorio** inadecuado de paradigmas de programación, un conocimiento deficiente de los paradigmas existentes, la manera en que enseñamos esos paradigmas y en la manera en que los lenguajes de programación soportan, o no, los paradigmas de sus comunidades de usuarios. Si el avance general del arte la programación requiere de la continua invención y elaboración de paradigmas, el avance individual requiere de la expansión del repertorio personal de paradigmas.

Paradigmas de Programación



Paradigmas de Programación

Existen muchos paradigmas pero los mas utilizados con:

- ▶ Programación estructurada
- ▶ Programación logica
- ▶ Programación funcional
- ▶ Programación orientada a objetos

Programación estructurada

- ▶ Agrupa las instrucciones en procedimientos, Lo que lo hace modular
- ▶ Tiene estructuras de control (ciclos, condicionales)
- ▶ Lenguajes:
 - ▶ C . Lenguaje de propósito General
 - ▶ Cobol (COmmon Business Oriented Language)
 - ▶ FORTRAN (FORmula TRANslation)
 - ▶ ALGOL (ALGOrithmic Language)
 - ▶ BASIC (Begginers All purpose Symbolic Instruction Code)
 - ▶ Pascal
 - ▶ Haskell

Programación lógica

- ▶ Basado en la deducción lógica
- ▶ Maneja hechos y relaciones
- ▶ Ampliamente utilizado en
 - ▶ demostración automática de teoremas
 - ▶ Sistemas expertos
 - ▶ Reconocimiento de lenguaje natural
- ▶ Lenguajes:
 - ▶ Prolog

Programación lógica

► Ejemplo:

Sean los axiomas:

- Todos los hombres son falibles.
- Socrates es un hombre.

Un teorema que lógicamente sigue de estos dos axiomas es:

- Socrates es falible.

El ejemplo puede entonces traducirse a Prolog como sigue:

1 **falible(X) :- hombre(X).**

2 **hombre(socrates).**

y

1 **?- falible(socrates)**

2 **true.**

Toda Generalización es mala

- ▶ El ron en las rocas emborracha
- ▶ El whisky en las rocas enborracaña
- ▶ El brandy en las rocas emborracha

Conclusión:

¡Las rocas emborran!!



Programación funcional

- ▶ Programación aplicable. En lugar de escribir programas, definimos funciones. En lugar de ejecutar programas, evaluamos expresiones.
- ▶ Sintaxis simplificada. En particular, la precedencia de los operadores es eliminada; y datos y programas comparten la misma sintaxis.
- ▶ Recursividad como la principal estructura de control.
- ▶ Lenguajes:
 - ▶ LISP
 - ▶ RUST
 - ▶ Scala
 - ▶ Haskell
- ▶ Lenguajes de propósito general que lo han incorporado:
 - Java PHP
 - Python SQL
 - C++
- ▶ Lenguajes de propósito específico que lo han incorporado:
 - R (a través de PURR)
 - Mathematica (Computo simbólico)
 - J y K (financiero)

Programación Funcional

- ▶ Ejemplos:

(+ 8 5)

; suma (8 y 5)

; Returns (13)

(sort (list 5 2 6 3 1 4) #'>)

; Ordena la lista usando la función > como el operador relacional.

; Returns (6 5 4 3 2 1).

(sort (list '(9 A) '(3 B) '(4 C)) #'< :key #'first)

; Ordena la lista de acuerdo al primer elemento de cada sublista.

; Returns ((3 B) (4 C) (9 A)).

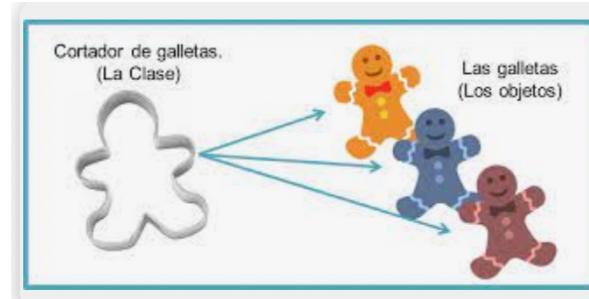
Programación Orientada a Objetos

- ▶ Se basa en el concepto de clase. Una clase es un conjunto de objeto que comparte un grupo de características (atributos) y funciones (metodos)
- ▶ Características:
 - ▶ Abstracción
 - ▶ Herencia
 - ▶ Polimorfismo
 - ▶ Encapsulamiento y principio de ocultación(Metodos publicos y privados)
 - ▶ Modularidad
- ▶ Lenguajes:

Smalltalk, C++, Java, R, Python, Ruby, Visual Object, Scala, Fortran95,..

Clases y Objetos en R

- ▶ Una clase es la definición de un conjunto con características y métodos
- ▶ Un objeto es una instancia de la clase



Clase: Cliente

Nombre

Id

Email

Teléfono

MostrarDatos()

Solicitar servicio()

Realizar pago()

Clase: Carro

Marca

Submarca

No. cilindros

Litros por km

color

Frenar()

Acelerar()

Girar()

Clase: Gene

Nombre

Pos Inicio

Posición Final

Secuencia

Porcentaje GC()

Transcribirse()

Generemos la clase silla...



Clases y Objetos en R



¿Cómo instalar R y como es el ambiente de trabajo de R Studio?

<https://www.youtube.com/> -> RLadiesCuernavaca

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays the following R code:

```
1 var1<-4
2
```
- Environment View:** Shows the variable `var1` with the value `4`.
- File View:** Shows a directory structure under `Home`, including files like `.Rhistory` and `Applications`, and folders like `bin`, `cursos`, `Desktop`, and `Documents`.
- Console View:** Displays the command `> var1<-4` and its result.

Material de la clase

https://github.com/RLadiesCuerna/meetup_enero_2023

Clases y Objetos en R

- ▶ Todas las variables en R, son objetos de una clase.

```
a<-5  
class(a)  
datos<-c(5,8,'izq','naranja')  
class(datos)  
datos  
lista<-list(5,TRUE,"naranja")  
class(lista)
```

Herencia

Clase: Cliente
Nombre
Id
Email
Telefono

Solicitar un servicio
Realizar un pago

Clase: ClienteUNAM
Dependencia
Solicitar un servicio
Realizar un pago

Clase: ClienteNoUNAM
Dependencia
Datos de facturacion
Solicitar un servicio
Realizar un pago
SolicitarFactura

Polimorfismo

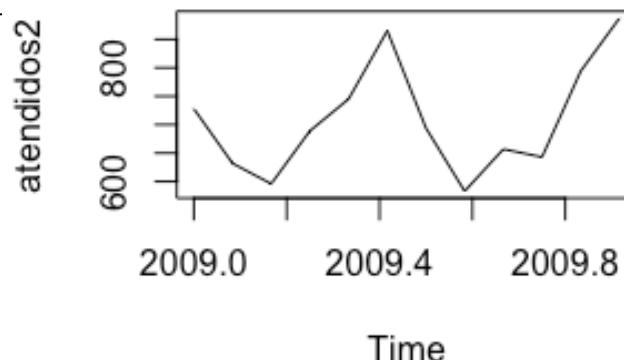
- ▶ Una misma función actua en función del tipo de parametros que recibe

```
> max(8,5,20)
[1] 20
> max("Maria","Juan","Mariana")
[1] "Mariana"
```

Encapsulamiento y principio de ocultación

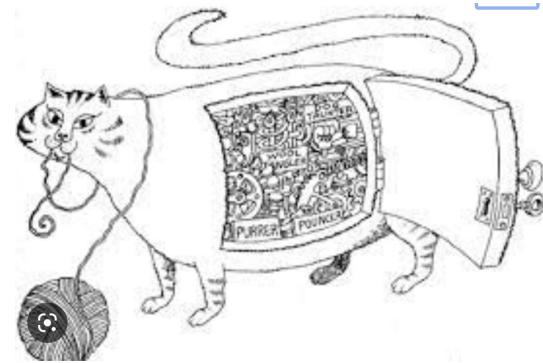
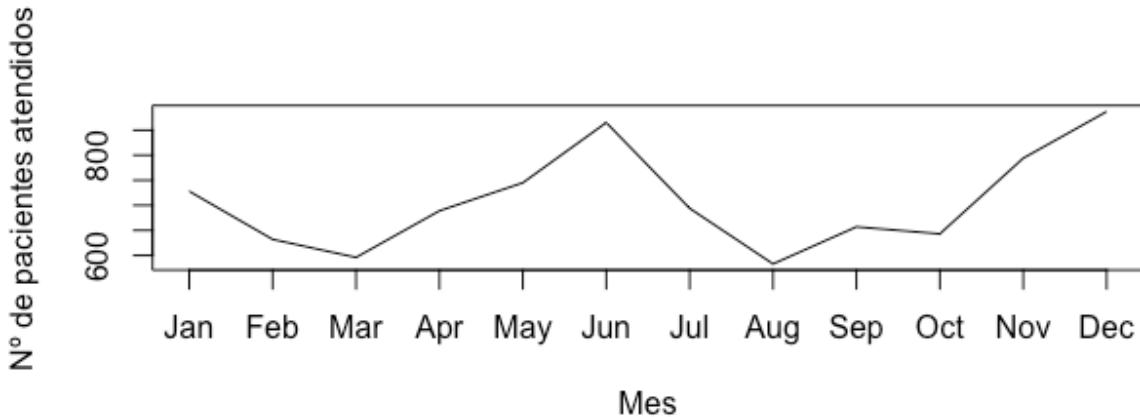
- ▶ Todas las variables en R, son objetos de una clase.

```
atendidos <- c(728,632,596,689,745,865,694,583,657,643,794,887)
class(atendidos)
help(ts)
atendidos2 <- ts(atendidos,frequency=12,start=c(2009,1))
atendidos2
class(atendidos2)
plot(atendidos2)
```



Encapsulamiento y principio de ocultación

```
# Generamos el gráfico sin las etiquetas del eje X (xaxt="n"):  
plot(atendidos2,xaxt="n",xlab="Mes",ylab="Pacientes atendidos")  
# Añadimos las etiquetas de los meses:  
axis(1,at=time(atendidos2),  
     labels=format(seq(as.Date("2009/1/1"),  
                      by="month", length=12),"%b"))
```



Modularidad



La modularidad empaquetada las abstracciones en unidades discretas.

Modularidad

R. contiene multiples paquetes para brindar conjuntos de herramientas con objetivos particulares

- ▶ Biostring
- ▶ ggplot
- ▶ stats
- ▶ ...

The screenshot shows the RStudio interface. On the left, the Environment panel is open, displaying the 'Packages' tab. It lists several packages installed in the current session, including base, Biobase, BiocGenerics, BiocManager, BiocVersion, and Biostrings. The 'base' package is checked as the active package. The code editor window on the right contains R code related to gene lists and GO ontologies.

Name	Description	Ver...
<input checked="" type="checkbox"/> base	The R Base Package	4.1.2
<input type="checkbox"/> base64enc	Tools for base64 encoding	0.1-3
<input type="checkbox"/> BH	Boost C++ Header Files	1.78.0-0
<input type="checkbox"/> Biobase	Biobase: Base functions for Bioconductor	2.54.0
<input type="checkbox"/> BiocGenerics	S4 generic functions used in Bioconductor	0.40.0
<input type="checkbox"/> BiocManager	Access the Bioconductor Project Package Repository	1.30.1
<input type="checkbox"/> BiocVersion	Set the appropriate version of Bioconductor packages	3.14.0
<input type="checkbox"/> Biostrings	Efficient manipulation of biological strings	2.62.0

```

Untitled1* >> [ ] [ ]
Source > [ ] [ ]
e,geneList, geneID2GO
IntologyType))
ls.MolecularFunction.o
escription="My project
: pueden ser accesados
elID2GO, OntologyType, myIn

```

<https://www.bioconductor.org>



Search:

[Home](#) [Install](#) [Help](#) [Developers](#) [About](#)

About *Bioconductor*

The mission of the *Bioconductor* project is to develop, support, and disseminate free open source software that facilitates rigorous and reproducible analysis of data from current and emerging biological assays. We are dedicated to building a diverse, collaborative, and welcoming community of developers and data scientists.

Bioconductor uses the R statistical programming language, and is open

Install »

- Discover [2183 software packages](#) available in *Bioconductor* release 3.16.

Get started with *Bioconductor*

- [Install *Bioconductor*](#)
- [Get support](#)
- [Latest newsletter](#)
- [Follow us on twitter](#)
- [Install R](#)

Learn »

Master *Bioconductor* tools

- [Courses](#)
- [Education and Training](#)
- [Support site](#)
- [Package vignettes](#)
- [Literature citations](#)
- [Common work flows](#)
- [FAQ](#)
- [Community resources](#)
- [Videos](#)
- [Bioconductor Community Blog](#)

<https://www.bioconductor.org>

¿Quieres crear tu propia clase?

- ▶ En R, tu puedes crear tus propias clases, con sus atributos y métodos usando las clases S3 y S4. Para más detalles revisar (<https://adv-r.hadley.nz>)

Veamos un ejemplo. Defina la clase cliente

```
setClass("cliente", slots=list(nombre="character", id="numeric",
                               email="character", tel="character"))
```

Ahora definimos un objeto (instancia) con la función new:

```
cliente1 <- new("cliente", nombre="Juan Ramon De la Fuente",
                 id=1002, email="jrfuente@unam.mx", tel="777-329-1777")
```

Modificando un atributo

- ▶ Podemos modificar el valor de un atributo usando el operador @

```
cliente1@email<- "jramon.fuentes@unam.mx"
cliente1
An object of class "cliente"
Slot "nombre":
[1] "Juan Ramon De la Fuente"

Slot "id":
[1] 1002

Slot "email":
[1] "jramon.fuentes@unam.mx"

Slot "tel":
[1] "777-329-1777"
```

Agregemos un metodo a la clase

- ▶ Redefinamos el metodo show() para mostrar los datos de un objeto de la clase cliente usando el metodo generico de S4 setMethod

```
setMethod("show",
  "cliente",
  function(object) {
    cat("Nombre:",object@nombre, "\n")
    cat("Id:",object@id, "\n")
    cat("email:", object@email, "\n")
    cat("telefono:",object@tel,"\\n")
  }
)
```

```
cliente1
Nombre: Juan Ramon De la Fuente
Id: 1002
email: jramon.fuentes@unam.mx
telefono: 777-329-1777
```

Resumen

- ▶ Revisamos los paradigmas de programación usados por R:
 - ▶ Programación estructurada
 - ▶ Programación funcional
 - ▶ Programación Orientada a Objetos
- ▶ Revisamos los conceptos de la POO:
 - ▶ Clase y Objeto; Herencia, polimorfismo, encapsulamiento y ocultación de datos y modularidad
- ▶ Vimos en el ambiente de R:
 - ▶ Clases, polimorfismo, ocultación de datos y modularidad
 - ▶ Como crear una clase y sus instancias u objetos usando S4

Referencias

- ▶ **Introducción a R.**
<https://synergy.vision/corpus/R/Intro-R/>
- ▶ **Programación para la Inteligencia Artificial**
Dr. Alejandro Guerra-Hernández. aguerra@uv.mx
(<https://www.uv.mx/personal/aguerra/files/2022/08/pia-01.pdf>)
- ▶ **The Structure of Scientific Revolutions** (1962; second edition 1970; third edition 1996; fourth edition 2012) .
Thomas S. Kuhn
- ▶ **Advance R.** Second Edition. 2019. **Hadley Wickham.**
Ed. Chapman and Hall/CRC. <https://adv-r.hadley.nz>

Gracias a
tod@s por su
atención

veronica.jimenez@ibt.unam.mx
cuernavaca@rladies.org



Rladies Cuernavaca



RLadiesCuerna

meetup Rladies Cuernavaca