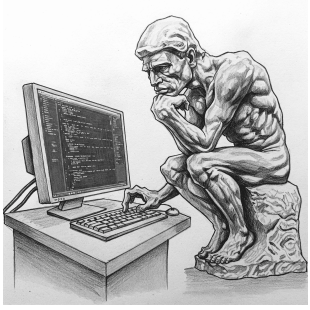
Análisis de datos en fortificación de alimentos a gran escala con R

Tema II: Importación de datos.

Dr. Maicel Monzón

Pregunta del encuentro anterior

Como se crea un objeto en R?



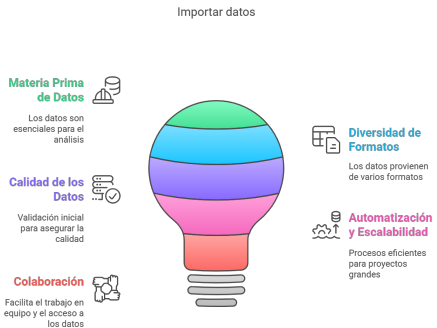
Respuesta

Los objetos se crean:

- Leyendo datos de un archivo
- Como resultado de un cálculo
- “Asignándoles un valor”
- etc.

Importar datos

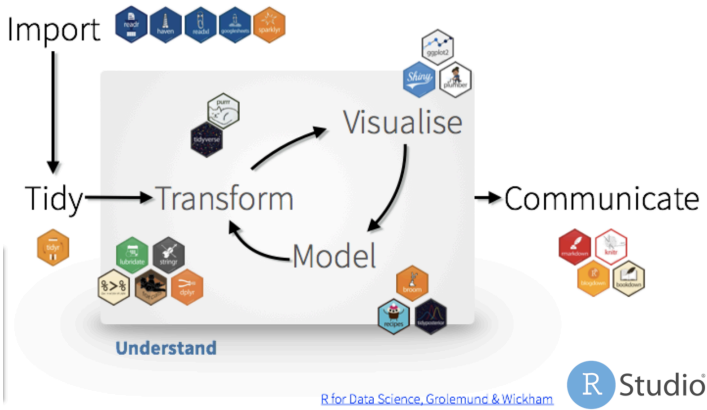
Vamos a hablar de una de las formas más usadas para **crear objetos** en R.



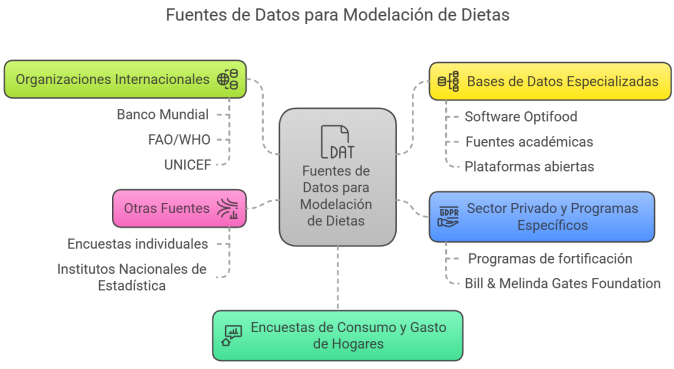
Sumario

Lectura de datos en el proceso de ciencia de datos

Importar datos: Es fundamental porque los datos son la base de cualquier análisis.



Fuentes de datos para la modelación de dietas



Qué datos se pueden leer desde R?

- **Datos estructurados** *(estructura interna identificable filas, columnas con títulos)*

Ej. archivos rectangulares en texto plano (csv,dat,txt)

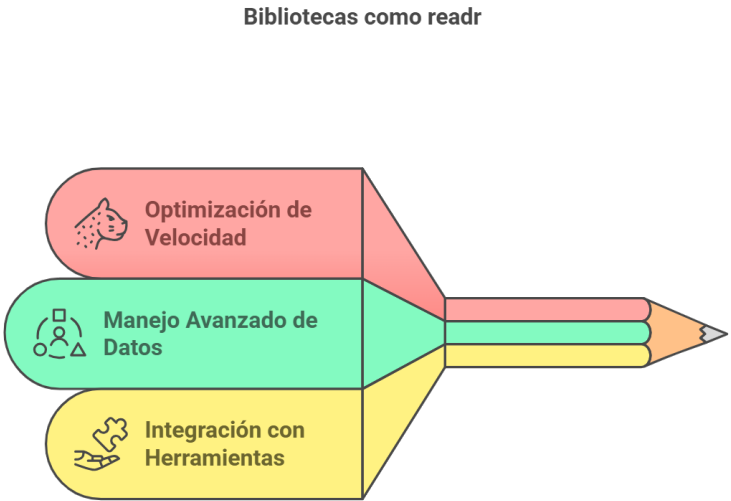
- **Datos no estructurados** *(datos binarios que no tienen estructura interna identificable)*

Ej. Correos electrónicos, publicaciones en redes sociales (Imágenes , sonido, video, etc.)

Bibliotecas especializadas en leer datos de distintos formatos

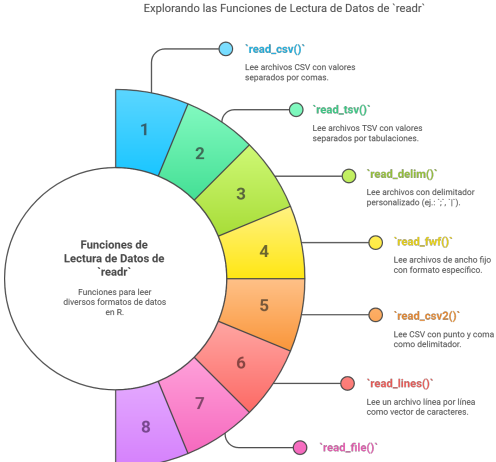
Paquete	formatos
readr*	datos rectangulares (csv, tsv y fwf)
haven	ficheros (SPSS, Stata y SAS)
readxl	ficheros Excel (.xls y .xlsx)
DBI	bases de datos
jsonlite	json
xml2	XML
httr	Web APIs
rvest	HTML (Web Scraping)

La biblioteca readr



Funciones para lectura de datos tabulares a tibbles (readr)

Estas funciones están diseñadas para leer datos rápidamente y devolver tibbles (data frames optimizados).



Delimitadores de readr

Funciones	delimitador
read_delim()	cualquier
read_csv()	coma
read_csv2()	punto y coma
read_tsv()	tabulaciones
read_fwf()	ancho fijo
fwf_widths()	ubicación

Algunas convenciones

path o camino (ruta hacia un archivo)

“./carpeta/archivo.ext”

separador entre carpetas es una barra inclinada (/),

(./) Espacio de trabajo

```
# obtiene el espacio de trabajo
ws <- getwd()
# contruye una ruta
file.path(".","datos","mifichero.csv")
```

Argumento file, readr(file)

```
 read_csv(
#file -cadena de texto- Argumento obligatorio
 file = "a,b,c \n
          1,2,3  \n
          4,5,6")
```

```
# A tibble: 2 x 3
      a     b     c
  <dbl> <dbl> <dbl>
1     1     2     3
2     4     5     6
```

Argumento skip, readr(file,skip)

```
read_csv(file = "La primer línea de metadata
  La segunda línea de metadata
  x,y,z
  1,2,3",
#skip -omitir las primeras n líneas-
  skip = 2,
  col_types ="i" )
```

Argumento col_names

```
read_csv(file ="41,masculino, blanca
         40, femenino, negra",
# incluye el nombres de columna -tipo booleano-
         col_names = FALSE)
```

Argumeto col_name -Asignación de nombres de columna-

```
read_csv(file ="41,masculino,blanca\n
         40, femenino,negra",
# col_names -vector de caracteres-
         col_names = c("edad","sexo","color de la piel"))
```

Argumento na -valores faltantes-

```
# carácter que se asume como valores faltantes, vector de cadena
read_csv(file ="a,b,c\n1,2, ", na = " ")
```

Parsing: proceso de convertir datos en formato correctos para su análisis

funciones parse_*()

funciones toman un vector de caracteres y devuelven:

- parse_logical() # lógico
- parse_integer() # entero
- parse_double() # decimal
- parse_number() # numérico
- parse_character() # cadena
- parse_factor() # tipo factor
- parse_date() # fecha
- parse_time() # tiempo

Parseo de números (Usos)

1. diferente de decimales *Ej coma o punto*

   `hemoglobina1<-c("12,2","13.5","11.9")` # diferente marca decimal

2. números están rodeados por otros caracteres *Ej unidades*

   `hemoglobina2<-c("12.2","13.5","11.9 g/l")` # texto adicionado

3. caracteres de “agrupación” *Ej “1,000,000”.*

Ejemplo 1: parse_double (locale) uso diferente de decimales

```
# locale -objeto que especifica las opciones de análisis que difieren de u
parse_double("1.23")
# decimal_mark "símbolo decimal"
parse_double("1,23",
             locale = locale(decimal_mark = ","))
```

Ejemplo 2: parse_number

```
# ignora los caracteres no-numéricos antes y después del númeroignora los c
parse_number("123 g/l")
parse_number("la hemoglobina es de 123 g/l")
```

Ejemplo 3 combinación de parse_number() y el locale

```
parse_number("123.456.789",
             locale = locale(grouping_mark = "."))
```

Parseo de Cadenas de caracteres (parse_character())

```
x1 <- "El Ni\xf1o"
# caracteres en español a veces no se leen bien ej ñ o acentos
parse_character(x1,
                locale = locale(encoding = "Latin1"))
```

Parseo de factores parse_factor()

```
fruta <- c("manzana", "banana")
# representar las variables categóricas que tienen un conjunto conocido de
parse_factor(c("manzana", "banana", "bananana"), levels = fruta)
```

Parseo de fechas (parse_date()

- Año : %y (2 dígitos)
- Mes : %m (2 dígitos).
- Mes : %b (nombre abreviado, como “ene”).
- Mes :: %B (nombre completo, “enero”).
- Dias : “%d” (2 dígitos)

Parseo de tiempos (parse_time())

```
parse_time("20:10:01")
```

Analizar automatico de un archivo con readr (guess_parser)

```
#  deduce automáticamente el formato con las primeras 1000 filas
guess_parser("2010-10-01")
guess_parser("15:01")
guess_parser(c("TRUE", "FALSE"))
guess_parser(c("1", "5", "9"))
guess_parser(c("12,352,561"))
guess_parser("12.3")
```

Detección automatica (1000 primeras filas)

guess_parser casi siempre detecta los tipos correctos, sin embargo, puede no ser así en todos los casos.

función problems(): Identificación de problemas de parseo

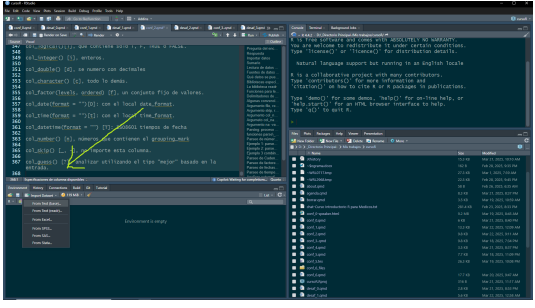
```
desafio <- read_csv("datos_fortificacion.csv")
# si sale notificaciones
problems(desafio)
```

Funciones para Definir Tipos de Columnas en readr

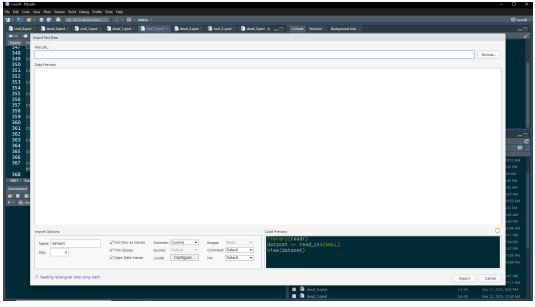
Cada función `col_*()` se usa para indicar el tipo de dato de una columna al importar datos.
Las **abreviaturas** (entre paréntesis) permiten definir tipos de forma rápida.

Función	Abreviatura	Descripción
`col_logical()`	`[l]`	**Valores lógicos:** `TRUE`, `FALSE`, `T`, `F`.
`col_integer()`	`[i]`	**Números enteros (e):** `-5`, `0`, `100`).
`col_double()`	`[d]`	**Números con decimales (e):** `3.14`, `-0.5`).
`col_character()`	`[c]`	**Texto (se usa si hay caracteres no numéricos). Valor por defecto en readr.**
`col_factor(levels, ordered)`	`[f]`	**Factores: categorías con niveles definidos (e): "hombre", "mujer").**
`col_date(format = "")`	`[D]`	**Fechas (e):** `"2023-12-25"`). **Usa el formato de fecha del** `locale`.
`col_time(format = "")`	`[t]`	**Horas (e):** `"14:30:00"`). **Usa el formato de hora del** `locale`.
`col_datetime(format = "")`	`[T]`	**Fechas y horas (e):** `"2023-12-25 14:30:00"`). **Formato ISO8601 por defecto.**
`col_number()`	`[n]`	**Números con separadores de miles (e):** `"1,000.50"` → `1000.5`).
`col_skip()`	`[_]` o `[-]`	**Omite la columna (no la importa).**
`col_guess()`	`[?]`	**Tipo automático (lo que hace readr por defecto si no se especifica).**

Importar con interfaz gráfica



Interfaz gráfica para la función readr



Conclusión

La **importación de datos** es el **primer paso crítico** en cualquier **análisis**. Con **readr** y **bibliotecas especializad**, R ofrece herramientas eficientes para manejar **múltiples formatos**. Dominar estos recursos permite **enfocarse en el análisis** y la **toma de decisiones basada en datos**.

¡Gracias por su atención!