

# Tipos de Datos: Raw, Technically Correct y Consistent

Tractament de les Dades - Universidad de Valencia

2026-02-04

## Contents

<b>1 Introducción</b>	<b>2</b>
<b>2 Caso de estudio: Tickets de Supermercado</b>	<b>2</b>
<b>3 1. Raw Data (Datos Brutos)</b>	<b>2</b>
3.1 Características de los datos brutos . . . . .	2
3.2 Estructura de los datos brutos . . . . .	3
<b>4 2. Technically Correct Data</b>	<b>3</b>
4.1 Proceso de corrección técnica . . . . .	3
4.2 Estructura después de corrección técnica . . . . .	4
4.3 Mejoras logradas . . . . .	4
<b>5 3. Consistent Data (Datos Consistentes)</b>	<b>4</b>
5.1 Proceso de estandarización . . . . .	4
5.2 Verificación de consistencia . . . . .	5
5.3 Mejoras adicionales logradas . . . . .	6
<b>6 Comparación Lado a Lado</b>	<b>6</b>
6.1 Ejemplo 1: Producto “Leche” . . . . .	6
6.2 Ejemplo 2: Fecha . . . . .	6
6.3 Ejemplo 3: Categoría . . . . .	7
6.4 Ejemplo 4: Cantidad . . . . .	7
<b>7 Visualización de Categorías</b>	<b>7</b>
<b>8 Resumen Conceptual</b>	<b>8</b>
<b>9 Diagrama de Flujo del Proceso</b>	<b>9</b>
<b>10 Funciones Útiles en R</b>	<b>9</b>
10.1 Conversión de tipos . . . . .	9
10.2 Limpieza de texto . . . . .	10
10.3 Validación . . . . .	10
<b>11 Ejercicio para Estudiantes</b>	<b>10</b>
<b>12 Conclusiones</b>	<b>11</b>

# 1 Introducción

En el proceso de tratamiento de datos, distinguimos tres niveles de calidad:

1. **Raw data (Datos brutos)**: Datos tal como salen del sistema, sin procesar
2. **Technically correct data (Datos técnicamente correctos)**: Tipos de datos correctos y formato válido
3. **Consistent data (Datos consistentes)**: Formato correcto + consistencia semántica + validación de reglas de negocio

Este documento ilustra las diferencias mediante un ejemplo práctico: **tickets de compra de supermercado**.

---

## 2 Caso de estudio: Tickets de Supermercado

Trabajaremos con datos de 5 tickets de compra para ilustrar cada nivel de calidad.

---

### 3 1. Raw Data (Datos Brutos)

Los datos brutos son tal como salen del sistema de caja, sin ningún procesamiento.

#### 3.1 Características de los datos brutos

- Múltiples formatos de fecha
- Espacios innecesarios
- Tipos de datos incorrectos (números almacenados como texto)
- Decimales con coma y punto mezclados
- Categorías inconsistentes (mayúsculas/minúsculas)
- Errores y valores inválidos sin tratar

```
# DATOS BRUTOS: tal como salen del sistema
raw_data <- data.frame(
  ticket_id = c("T001", "T002", "T003", "T004", "T005"),
  fecha = c("2024-01-15", "15/01/2024", "2024.01.16", "16-Ene-24", "ERROR"),
  producto = c(" LECHE ", "Pan integral", "MANZANAS", "Yogur ", "aceite oliva"),
  cantidad = c("2", "1.5", "3", "6", "dos"),
  precio_unitario = c("1.20", "2,50", "1.80", "0.80", "4.5"),
  total = c("2.40", "3.75", "5,40", "4.8", "ERROR"),
  categoria = c("lacteos", "PANADERIA", "frutas", "Lácteos", "Aceites"),
  stringsAsFactors = FALSE
)

print(raw_data)
```

	ticket_id	fecha	producto	cantidad	precio_unitario	total	categoria
1	T001	2024-01-15	LECHE	2	1.20	2.40	lacteos
2	T002	15/01/2024	Pan integral	1.5	2,50	3.75	PANADERIA
3	T003	2024.01.16	MANZANAS	3	1.80	5,40	frutas
4	T004	16-Ene-24	Yogur	6	0.80	4.8	Lácteos
5	T005	ERROR	aceite oliva	dos	4.5	ERROR	Aceites

### 3.2 Estructura de los datos brutos

```
str(raw_data)

'data.frame': 5 obs. of 7 variables:
$ ticket_id      : chr  "T001" "T002" "T003" "T004" ...
$ fecha          : chr  "2024-01-15" "15/01/2024" "2024.01.16" "16-Ene-24" ...
$ producto       : chr  "LECHE" "Pan integral" "MANZANAS" "Yogur"   ...
$ cantidad        : chr  "2" "1.5" "3" "6" ...
$ precio_unitario: chr  "1.20" "2,50" "1.80" "0.80" ...
$ total           : chr  "2.40" "3.75" "5,40" "4.8" ...
$ categoria       : chr  "lacteos" "PANADERIA" "frutas" "Lácteos" ...
```

**Observación:** Todas las columnas son de tipo `character` (texto), incluso las que deberían ser numéricas o fechas.

---

## 4 2. Technically Correct Data

Datos que tienen el **tipo correcto** y **formato válido**, pero aún pueden tener inconsistencias semánticas.

### 4.1 Proceso de corrección técnica

```
# Iniciar con datos brutos
datos_tecnicos <- raw_data

# 1. CONVERTIR FECHAS
# Limpiar y estandarizar diferentes formatos de fecha
datos_tecnicos$fecha <- as.Date(ifelse(
  datos_tecnicos$fecha == "ERROR",
  NA,
  as.character(as.Date(datos_tecnicos$fecha, tryFormats = c("%Y-%m-%d", "%d/%m/%Y", "%Y.%m.%d"))),
  origin = "1970-01-01"))

# 2. ELIMINAR ESPACIOS
datos_tecnicos$producto <- trimws(datos_tecnicos$producto)

# 3. CONVERTIR CANTIDAD A NUMÉRICO
# Reemplazar texto por números
datos_tecnicos$cantidad <- ifelse(datos_tecnicos$cantidad == "dos", "2", datos_tecnicos$cantidad)
datos_tecnicos$cantidad <- as.numeric(datos_tecnicos$cantidad)

# 4. CONVERTIR PRECIO A NUMÉRICO
# Reemplazar comas por puntos
datos_tecnicos$precio_unitario <- as.numeric(gsub(", ", ".", datos_tecnicos$precio_unitario))

# 5. CONVERTIR TOTAL A NUMÉRICO
datos_tecnicos$total <- ifelse(datos_tecnicos$total == "ERROR", NA, datos_tecnicos$total)
datos_tecnicos$total <- as.numeric(gsub(", ", ".", datos_tecnicos$total))

# Mostrar resultado
print(datos_tecnicos)
```

ticket_id	fecha	producto	cantidad	precio_unitario	total	categoria
T001	2024-01-15	LECHE	2	1.20	2.40	Lácteos
T002	15/01/2024	Pan integral	1.5	2,50	3.75	PANADERIA
T003	2024.01.16	MANZANAS	3	1.80	5,40	frutas
T004	16-Ene-24	Yogur	6	0.80	4.8	Lácteos

1	T001	2024-01-15	LECHE	2.0	1.2	2.40	lacteos
2	T002	<NA>	Pan integral	1.5	2.5	3.75	PANADERIA
3	T003	<NA>	MANZANAS	3.0	1.8	5.40	frutas
4	T004	<NA>	Yogur	6.0	0.8	4.80	Lácteos
5	T005	<NA>	aceite oliva	2.0	4.5	NA	Aceites

## 4.2 Estructura después de corrección técnica

```
str(datos_tecnicos)
```

```
'data.frame': 5 obs. of 7 variables:
 $ ticket_id    : chr "T001" "T002" "T003" "T004" ...
 $ fecha         : Date, format: "2024-01-15" NA ...
 $ producto      : chr "LECHE" "Pan integral" "MANZANAS" "Yogur" ...
 $ cantidad      : num 2 1.5 3 6 2
 $ precio_unitario: num 1.2 2.5 1.8 0.8 4.5
 $ total          : num 2.4 3.75 5.4 4.8 NA
 $ categoria      : chr "lacteos" "PANADERIA" "frutas" "Lácteos" ...
```

## 4.3 Mejoras logradas

MEJORAS:

- \* Fechas convertidas a tipo Date
- \* Espacios eliminados
- \* Cantidades y precios como numeric
- \* Decimales uniformizados (punto decimal)
- \* Valores inválidos convertidos a NA

PROBLEMAS PENDIENTES:

- \* Nombres de productos inconsistentes (MAYÚSCULAS/minúsculas)
- \* Categorías con duplicados semánticos ('lacteos' vs 'Lácteos')
- \* No hay validación de reglas de negocio

## 5 3. Consistent Data (Datos Consistentes)

Datos con formato correcto Y **consistencia semántica**, siguiendo reglas de negocio y convenciones uniformes.

### 5.1 Proceso de estandarización

```
# Partir de datos técnicamente correctos
datos_consistentes <- datos_tecnicos

# 1. ESTANDARIZAR NOMBRES DE PRODUCTOS (Title Case)
datos_consistentes$producto <- tools::toTitleCase(tolower(datos_consistentes$producto))

# 2. ESTANDARIZAR CATEGORÍAS
# Crear mapeo de categorías uniformes
```

```

categorias_map <- c(
  "lacteos" = "Lácteos",
  "Lácteos" = "Lácteos",
  "PANADERIA" = "Panadería",
  "frutas" = "Frutas",
  "Aceites" = "Aceites"
)
datos_consistentes$categoría <- categorias_map[datos_consistentes$categoría]

# 3. VALIDAR REGLAS DE NEGOCIO
# Regla: total = cantidad × precio_unitario
datos_consistentes$total_calculado <- datos_consistentes$cantidad *
  datos_consistentes$precio_unitario

# Verificar si los totales coinciden (tolerancia de 0.01 por redondeo)
datos_consistentes$total_correcto <- abs(datos_consistentes$total -
  datos_consistentes$total_calculado) < 0.01

# Mostrar resultado
print(datos_consistentes)

```

	ticket_id	fecha	producto	cantidad	precio_unitario	total	categoría
1	T001	2024-01-15	Leche	2.0	1.2	2.40	Lácteos
2	T002	<NA>	Pan Integral	1.5	2.5	3.75	Panadería
3	T003	<NA>	Manzanas	3.0	1.8	5.40	Frutas
4	T004	<NA>	Yogur	6.0	0.8	4.80	Lácteos
5	T005	<NA>	Aceite Oliva	2.0	4.5	NA	Aceites
			total_calculado	total_correcto			
1			2.40	TRUE			
2			3.75	TRUE			
3			5.40	TRUE			
4			4.80	TRUE			
5			9.00	NA			

## 5.2 Verificación de consistencia

```
cat("VALIDACIÓN DE REGLAS DE NEGOCIO:\n")
```

VALIDACIÓN DE REGLAS DE NEGOCIO:

```
cat("-----\n\n")
```

```
# Verificar que todos los totales sean correctos
totales_validos <- all(datos_consistentes$total_correcto, na.rm = TRUE)
cat("Todos los totales son correctos:", totales_validos, "\n\n")
```

Todos los totales son correctos: TRUE

```
# Mostrar detalles de la verificación
cat("Detalles por ticket:\n")
```

Detalles por ticket:

```

for(i in 1:nrow(datos_consistentes)) {
  if(!is.na(datos_consistentes$total_correcto[i])) {
    cat(sprintf("Ticket %s: %.2f × %.2f = %.2f [%s]\n",
                datos_consistentes$ticket_id[i],
                datos_consistentes$cantidad[i],
                datos_consistentes$precio_unitario[i],
                datos_consistentes$total_calculado[i],
                ifelse(datos_consistentes$total_correcto[i], "OK", "ERROR")))
  }
}

```

Ticket T001: 2.00 × 1.20 = 2.40 [OK]  
 Ticket T002: 1.50 × 2.50 = 3.75 [OK]  
 Ticket T003: 3.00 × 1.80 = 5.40 [OK]  
 Ticket T004: 6.00 × 0.80 = 4.80 [OK]

### 5.3 Mejoras adicionales logradas

MEJORAS ADICIONALES:

- \* Nombres en formato Title Case uniforme
  - \* Categorías estandarizadas (sin duplicados semánticos)
  - \* Validación de reglas de negocio implementada
  - \* Convenciones uniformes en todo el dataset
  - \* Trazabilidad de errores (columna total\_correcto)
- 

## 6 Comparación Lado a Lado

### 6.1 Ejemplo 1: Producto “Leche”

EVOLUCIÓN DEL PRODUCTO 'LECHE':

```
=====
Raw data:          ' LECHE ' (con espacios, mayúsculas)
Technically correct: 'LECHE' (sin espacios, aún en mayúsculas)
Consistent data:      'Leche' (Title Case estandarizado)
```

### 6.2 Ejemplo 2: Fecha

EVOLUCIÓN DE LA FECHA:

```
=====
Raw data:          '15/01/2024' (texto, formato variable)
Technically correct: 2024-01-15 (tipo Date)
Consistent data:      2024-01-15 (tipo Date + validado)
```

### 6.3 Ejemplo 3: Categoría

EVOLUCIÓN DE CATEGORÍAS:

```
=====
Raw data:          'lacteos', 'Lácteos' (duplicados semánticos)
Technically correct:  'lacteos', 'Lácteos' (aún inconsistente)
Consistent data:    'Lácteos' (estandarizado, sin duplicados)
```

### 6.4 Ejemplo 4: Cantidad

EVOLUCIÓN DE CANTIDAD:

```
=====
Raw data:          'dos' (texto)
Technically correct:  2.0 (numérico)
Consistent data:    2.0 (numérico + validado con reglas)
```

## 7 Visualización de Categorías

```
# Crear gráfico comparativo de categorías
par(mfrow = c(1, 3), mar = c(10, 4, 4, 2))

# Raw data - categorías originales
tabla_raw <- table(raw_data$categoría)
barplot(tabla_raw,
        main = "Raw Data",
        ylab = "Frecuencia",
        col = "lightcoral",
        las = 2,
        cex.names = 0.8)

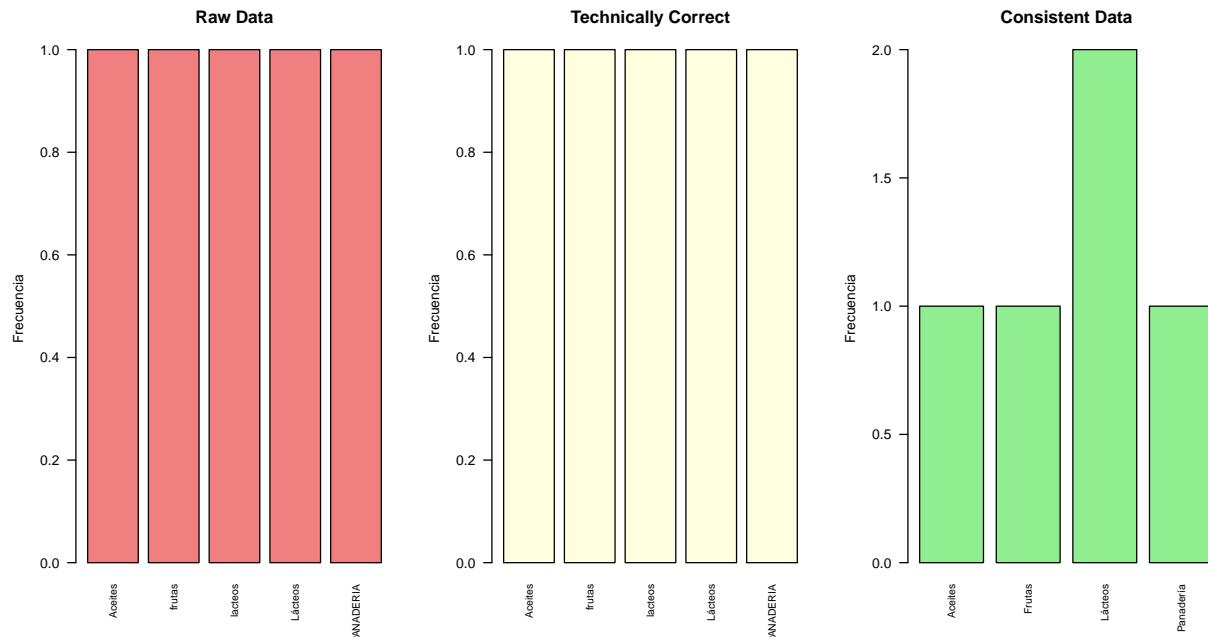
# Technically correct - mismas inconsistencias
tabla_tech <- table(datos_técnicos$categoría)
barplot(tabla_tech,
        main = "Technically Correct",
        ylab = "Frecuencia",
        col = "lightyellow",
        las = 2,
        cex.names = 0.8)

# Consistent - categorías estandarizadas
tabla_cons <- table(datos_consistentes$categoría)
barplot(tabla_cons,
        main = "Consistent Data",
        ylab = "Frecuencia",
        col = "lightgreen",
```

```

    las = 2,
    cex.names = 0.8)

```



**Observación:** Solo en los datos consistentes vemos que “Lácteos” aparece 2 veces (correctamente agrupados).

---

## 8 Resumen Conceptual

---

### RESUMEN DE NIVELES DE CALIDAD

---

#### 1. RAW DATA (Datos Brutos)

- \* Como salen del sistema
- \* Múltiples formatos
- \* Tipos mezclados
- \* Errores sin tratar
- \* Inconsistencias

#### 2. TECHNICALLY CORRECT DATA

- \* Tipos de datos correctos
- \* Fechas como Date
- \* Números como numeric

- \* NA para valores inválidos
- \* Sin espacios extras
- \* Aún puede haber inconsistencias semánticas

### 3. CONSISTENT DATA

- \* Todo lo anterior +
  - \* Convenciones uniformes
  - \* Categorías estandarizadas
  - \* Reglas de negocio validadas
  - \* Semántica coherente
  - \* Listo para análisis
- 

## 9 Diagrama de Flujo del Proceso

RAW DATA              Como sale del sistema  
data.frame            Todos los campos como character

1. Conversión de tipos
2. Limpieza de espacios
3. Tratamiento de errores

TECHNICALLY CORRECT      Tipos correctos  
data.frame                Formato válido

1. Estandarización de nombres
2. Unificación de categorías
3. Validación de reglas

CONSISTENT DATA            Listo para análisis  
data.frame                Calidad garantizada

---

## 10 Funciones Útiles en R

### 10.1 Conversión de tipos

```
# Convertir a Date
as.Date("2024-01-15")
as.Date("15/01/2024", format = "%d/%m/%Y")
```

```

# Convertir a numérico
as.numeric("123.45")
as.numeric(gsub(", ", ".", "123,45")) # Reemplazar coma por punto

# Convertir a lógico
as.logical(c(1, 0, TRUE, FALSE))

```

## 10.2 Limpieza de texto

```

# Eliminar espacios
trimws(" texto ")

# Cambiar mayúsculas/minúsculas
toupper("texto")           # TODO MAYÚSCULAS
tolower("TEXTO")           # todo minúsculas
tools::toTitleCase("texto") # Primera Letra Mayúscula

# Reemplazar caracteres
gsub(", ", ".", "1,23")    # Reemplazar todas las comas
sub(", ", ".", "1,23")     # Reemplazar primera coma

```

## 10.3 Validación

```

# Verificar valores faltantes
is.na(x)
sum(is.na(x))

# Verificar duplicados
duplicated(x)
unique(x)

# Rango de valores
range(x, na.rm = TRUE)

```

# 11 Ejercicio para Estudiantes

Aplica el proceso completo a este nuevo conjunto de datos:

```

# Datos brutos de ventas
ventas_raw <- data.frame(
  id = c("V001", "V002", "V003", "V004"),
  fecha = c("2024-02-01", "01/02/2024", "2024.02.02", "ERROR"),
  cliente = c(" JUAN PÉREZ ", "maría garcía", "PEDRO lópez", "Ana Torres "),
  monto = c("150.50", "200,75", "PENDIENTE", "99.99"),
  estado = c("PAGADO", "pagado", "Pendiente", "PAGADO"),
  stringsAsFactors = FALSE
)

# TAREAS:
# 1. Convertir a "Technically Correct"
# 2. Convertir a "Consistent Data"

```

```
# 3. Validar que los estados sean solo "Pagado" o "Pendiente"  
# 4. Crear una columna que indique si el monto es mayor a 100  
  
# Tu código aquí...
```

---

## 12 Conclusiones

- **Raw data** es el punto de partida, con todos los problemas típicos de datos reales
- **Technically correct** resuelve problemas técnicos de tipos y formatos
- **Consistent data** añade consistencia semántica y validación de negocio
- Solo los **datos consistentes** están listos para análisis estadístico confiable

**Regla de oro:** Nunca analices datos sin pasar por estos tres niveles.

---

**Fin del documento**