Estandarización vs Normalización de Datos

Tractament de les Dades - Universidad de Valencia

2026-02-04

Contents

1 Introducción

En el proceso de **limpieza y procesamiento de datos**, es fundamental distinguir entre dos conceptos que a menudo se confunden:

- **Estandarización de datos (sentido amplio)**: Proceso completo de transformar TODOS los datos a un formato coherente y utilizable
- **Normalización/Escalado (sentido específico)**: Operación específica sobre variables numéricas para llevarlas a una escala común

La **estandarización amplia** INCLUYE la normalización como uno de sus pasos, pero va mucho más allá.

---

2 Datos de ejemplo

Creamos un conjunto de datos con problemas típicos de formato:

```
# Datos brutos con problemas de formato
datos_brutos <- data.frame(
  nombre = c("Ana García", "PEDRO lópez", "  María Torres", "juan Pérez  ", "Lucía Martín"),
  edad = c(22, 19, 25, NA, 23),
  horas_estudio = c(15, 8, 20, 12, 18),
  nivel = c("Grado", "grado", "MÁSTER", "  Grado ", "Postgrado"),
  ciudad = c("Valencia", "madrid", "BARCELONA", "valencia ", "  Madrid"),
  beca = c("Sí", "No", "si", "NO", "SÍ"),
  nota_media = c(7.5, 5.2, 8.9, 6.1, 7.8),
  stringsAsFactors = FALSE
)

print(datos_brutos)
```

```
          nombre edad horas_estudio     nivel    ciudad beca nota_media
1     Ana García   22            15     Grado  Valencia   Sí        7.5
2    PEDRO lópez   19             8     grado    madrid   No        5.2
3   María Torres   25            20    MÁSTER BARCELONA   si        8.9
4   juan Pérez     NA            12    Grado  valencia    NO        6.1
5   Lucía Martín   23            18 Postgrado    Madrid   SÍ        7.8
```

Problemas identificados:

- Espacios en blanco innecesarios
- Inconsistencia en mayúsculas/minúsculas
- Valores categóricos no uniformes
- Valores faltantes (NA)

---

3 Paso 1: Corrección de Formatos

3.1 Eliminar espacios en blanco

```
# Eliminar espacios al inicio y final
datos_brutos$nombre <- trimws(datos_brutos$nombre)
datos_brutos$nivel <- trimws(datos_brutos$nivel)
datos_brutos$ciudad <- trimws(datos_brutos$ciudad)
```

3.2 Uniformizar mayúsculas/minúsculas

```
# Nombres propios: Primera letra mayúscula
datos_brutos$nombre <- tools::toTitleCase(tolower(datos_brutos$nombre))
datos_brutos$ciudad <- tools::toTitleCase(tolower(datos_brutos$ciudad))

# Categorías: Capitalizar primera letra
```

```r
datos_brutos$nivel <- paste0(toupper(substring(datos_brutos$nivel, 1, 1)),
                             tolower(substring(datos_brutos$nivel, 2)))
```

3.3 Estandarizar valores categóricos

```r
# Estandarizar valores booleanos
datos_brutos$beca <- tolower(datos_brutos$beca)
datos_brutos$beca[datos_brutos$beca %in% c("sí", "si")] <- "Sí"
datos_brutos$beca[datos_brutos$beca == "no"] <- "No"
```

3.4 Resultado después de corrección

```r
print(datos_brutos)
```

```
        nombre edad horas_estudio     nivel    ciudad beca nota_media
1   Ana García   22            15     Grado  Valencia   Sí        7.5
2  Pedro López   19             8     Grado    Madrid   No        5.2
3 María Torres   25            20    Máster Barcelona   Sí        8.9
4   Juan Pérez   NA            12     Grado  Valencia   No        6.1
5 Lucía Martín   23            18 Postgrado    Madrid   Sí        7.8
```

---

4 Paso 2: Imputación de Valores Faltantes

```r
# Rellenar valores faltantes con la mediana
datos_brutos$edad[is.na(datos_brutos$edad)] <- median(datos_brutos$edad, na.rm = TRUE)

print("Después de imputación:")
```

```
[1] "Después de imputación:"
```

```r
print(datos_brutos)
```

```
        nombre edad horas_estudio     nivel    ciudad beca nota_media
1   Ana García 22.0            15     Grado  Valencia   Sí        7.5
2  Pedro López 19.0             8     Grado    Madrid   No        5.2
3 María Torres 25.0            20    Máster Barcelona   Sí        8.9
4   Juan Pérez 22.5            12     Grado  Valencia   No        6.1
5 Lucía Martín 23.0            18 Postgrado    Madrid   Sí        7.8
```

Nota: La mediana es más robusta que la media ante valores atípicos.

---

5 Paso 3: Codificación de Variables Categóricas

5.1 Variable booleana (Sí/No → 1/0)

```r
datos_brutos$beca_num <- ifelse(datos_brutos$beca == "Sí", 1, 0)
```

5.2 Variable ordinal (con orden jerárquico)

```
# Grado < Postgrado < Máster
nivel_orden <- c("Grado" = 1, "Postgrado" = 2, "Máster" = 3)
datos_brutos$nivel_num <- nivel_orden[datos_brutos$nivel]
```

5.3 Variable nominal (One-Hot Encoding)

```
# One-Hot Encoding para ciudad
datos_brutos$ciudad_Barcelona <- ifelse(datos_brutos$ciudad == "Barcelona", 1, 0)
datos_brutos$ciudad_Madrid <- ifelse(datos_brutos$ciudad == "Madrid", 1, 0)
datos_brutos$ciudad_Valencia <- ifelse(datos_brutos$ciudad == "Valencia", 1, 0)
```

5.4 Resultado después de codificación

```
print(datos_brutos)
```

```
        nombre edad horas_estudio     nivel    ciudad beca nota_media beca_num
1   Ana García 22.0            15     Grado  Valencia   Sí        7.5        1
2  Pedro López 19.0             8     Grado    Madrid   No        5.2        0
3 María Torres 25.0            20    Máster Barcelona   Sí        8.9        1
4   Juan Pérez 22.5            12     Grado  Valencia   No        6.1        0
5 Lucía Martín 23.0            18 Postgrado    Madrid   Sí        7.8        1
  nivel_num ciudad_Barcelona ciudad_Madrid ciudad_Valencia
1         1                0             0               1
2         1                0             1               0
3         3                1             0               0
4         1                0             0               1
5         2                0             1               0
```

---

6 Paso 4: Normalización (Sentido Específico)

Solo aplicamos normalización a **variables numéricas continuas**.

6.1 Variables a normalizar

```
columnas_num <- c("edad", "horas_estudio", "nota_media")
```

6.2 Método 1: Min-Max Normalization (rango 0-1)

Fórmula:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```
# Función para normalización Min-Max
min_max_norm <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# Aplicar normalización
datos_minmax <- datos_brutos
```

```r
datos_minmax$edad <- min_max_norm(datos_brutos$edad)
datos_minmax$horas_estudio <- min_max_norm(datos_brutos$horas_estudio)
datos_minmax$nota_media <- min_max_norm(datos_brutos$nota_media)

print("Normalización Min-Max (0-1):")
```

```
[1] "Normalización Min-Max (0-1):"
```

```r
print(datos_minmax[, columnas_num])
```

```
       edad horas_estudio nota_media
1 0.5000000     0.5833333  0.6216216
2 0.0000000     0.0000000  0.0000000
3 1.0000000     1.0000000  1.0000000
4 0.5833333     0.3333333  0.2432432
5 0.6666667     0.8333333  0.7027027
```

Interpretación: Todos los valores están ahora entre 0 y 1, donde 0 es el mínimo original y 1 es el máximo original.

6.3 Método 2: Estandarización Z-score (media=0, sd=1)

Fórmula:

$$X_{std} = \frac{X - \mu}{\sigma}$$

```r
# Estandarización Z-score
datos_zscore <- datos_brutos
datos_zscore$edad <- scale(datos_brutos$edad)
datos_zscore$horas_estudio <- scale(datos_brutos$horas_estudio)
datos_zscore$nota_media <- scale(datos_brutos$nota_media)

print("Estandarización Z-score:")
```

```
[1] "Estandarización Z-score:"
```

```r
print(datos_zscore[, columnas_num])
```

```
         edad horas_estudio nota_media
1 -0.13837968    0.08377078  0.2743977
2 -1.52217649   -1.38221790 -1.3033892
3  1.24541713    1.13090555  1.2347898
4  0.09225312   -0.54451008 -0.6859943
5  0.32288592    0.71205164  0.4801960
```

6.4 Verificación de Z-score

```r
cat("\nMedias (deben ser aprox. 0):\n")
```

```

Medias (deben ser aprox. 0):
```

```r
print(colMeans(datos_zscore[, columnas_num]))
```

```
         edad horas_estudio    nota_media
-3.191891e-16  6.661338e-17  2.664535e-16
```

```r
cat("\nDesviaciones estándar (deben ser = 1):\n")
```

```
Desviaciones estándar (deben ser = 1):
```

```r
print(apply(datos_zscore[, columnas_num], 2, sd))
```

```
         edad horas_estudio    nota_media
            1             1             1
```

---

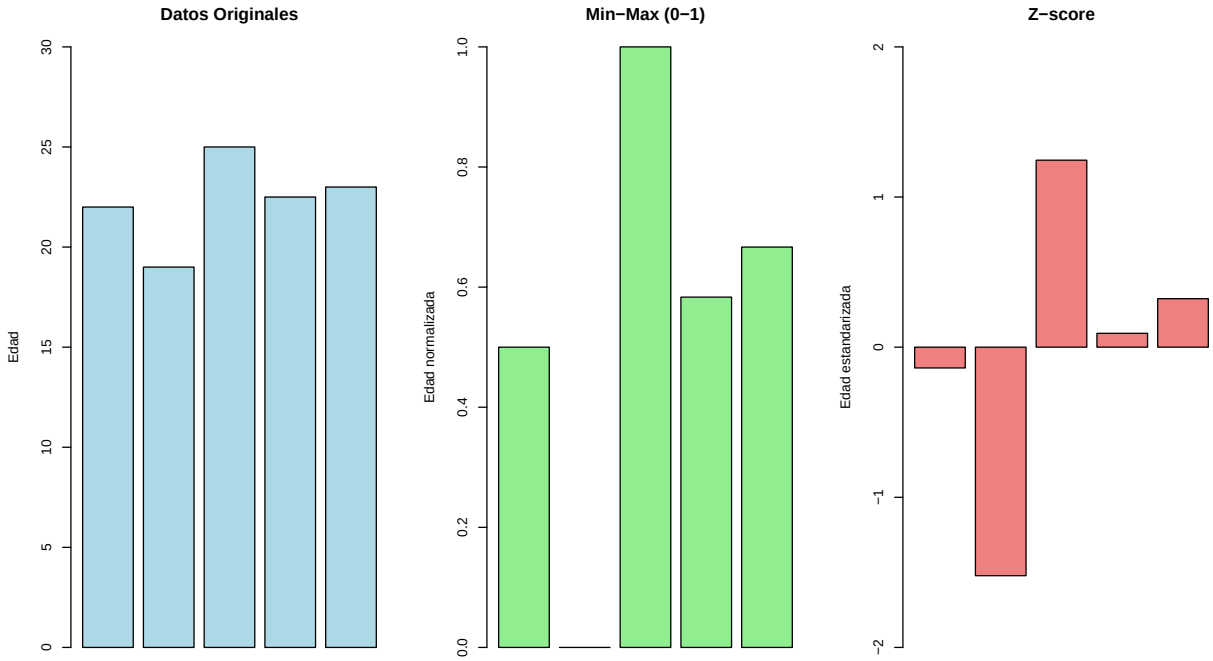
7 Comparación Visual

```r
# Configurar gráficos
par(mfrow = c(1, 3), mar = c(5, 4, 4, 2))

# Datos originales
barplot(datos_brutos$edad,
        main = "Datos Originales",
        ylab = "Edad",
        col = "lightblue",
        ylim = c(0, 30))

# Min-Max
barplot(datos_minmax$edad,
        main = "Min-Max (0-1)",
        ylab = "Edad normalizada",
        col = "lightgreen",
        ylim = c(0, 1))

# Z-score
barplot(as.numeric(datos_zscore$edad),
        main = "Z-score",
        ylab = "Edad estandarizada",
        col = "lightcoral",
        ylim = c(-2, 2))
```



---

8 Resumen del Proceso

```
========================================
PROCESO COMPLETO DE ESTANDARIZACIÓN
========================================
1. CORRECCIÓN DE FORMATOS:
   * Eliminación de espacios
   * Uniformización mayúsculas/minúsculas
   * Estandarización de valores categóricos
2. IMPUTACIÓN:
   * Valores faltantes → mediana
3. CODIFICACIÓN:
   * Booleanas: Sí/No → 1/0
   * Ordinales: Grado(1), Postgrado(2), Máster(3)
   * Nominales: One-Hot Encoding
4. NORMALIZACIÓN (sentido específico):
   * Min-Max: valores entre 0 y 1
   * Z-score: media=0, desviación=1
========================================
```

9 Cuándo usar cada método

Método	Cuándo usar	Ventajas	Desventajas
Min-Max	Límites conocidos, sin outliers	Interpretación intuitiva (0-1)	Sensible a outliers
Z-score	Distribución normal, hay outliers	Robusto a outliers	Valores sin límite definido
Robust Scaling	Muchos outliers	Muy robusto	Menos común

10 Referencias

- **Funciones R base útiles:**
  - `trimws()`: eliminar espacios
  - `toupper()`, `tolower()`, `tools::toTitleCase()`: mayúsculas/minúsculas
  - `scale()`: estandarización Z-score
  - `ifelse()`: codificación condicional
- **Documentación:**
  - `?scale`
  - `?trimws`