Transformando Datos en Estadística R: Fundamentos para el Análisis

Realizado por Juan Pablo Ramirez Correa

UNICISO
WWW.PORTALUNICISO.COM

UNICISO
WWW.PORTALUNICISO.COM

Tabla de Contenido

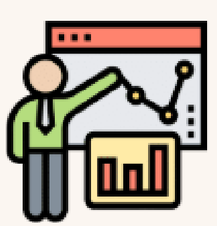
UNICISO
WWW.PORTALUNICISO.COM

¿Por qué es importante la transformación de datos?

La transformación de datos es un paso esencial en cualquier análisis. Permite limpiar, organizar y estructurar la información para facilitar su interpretación. **Los datos en bruto suelen contener errores, valores faltantes o formatos inconsistentes que pueden afectar los resultados del análisis.**

Permite limpiar, organizar y estructurar la información.

Facilita la visualización y exploración de datos.



Mejora la interpretación y precisión del análisis.

Corrige problemas como valores faltantes o formatos inconsistentes.

"**La transformación de datos garantiza la compatibilidad con los sistemas de destino y mejora la calidad y la usabilidad de los datos**. Es un aspecto esencial de las prácticas de gestión de datos, incluida la disputa de datos, el análisis de datos y el almacenamiento de datos" (IBM, 2024).

UNICISO
WWW.PORTALUNICISO.COM

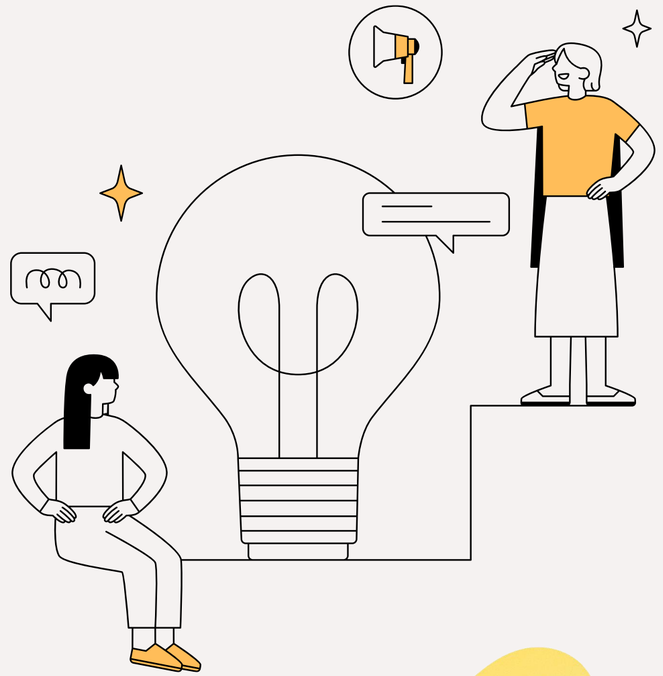
Introducción a los Paquetes Esenciales

En R, existen paquetes especializados que facilitan la transformación de datos. Siendo los más utilizados en ciencias sociales: **dplyr, tidyr y readr**, los cuales forman parte del ecosistema **tidyverse**.

dplyr Manipulación eficiente de datos (filtrar, seleccionar, mutar, resumir).

tidyr: Reorganización de datos (pivotar, separar, unir columnas).

readr: Importación de datos desde archivos externos (CSV, TXT, Excel).



Para instalar todo el ecosistema **tidyverse** se utiliza el siguiente comando, dentro de la ventana de instrucciones (script de Rstudio):

```
install.packages("tidyverse")
library(tidyverse)
```

Cuadro de ejemplo

UNICISO
WWW.PORTALUNICISO.COM

Carga y Exploración de Datos

En R, podemos importar datos desde **diversas fuentes (CSV, Excel, TXT, JSON, HTML/XML).** Para ello, se utilizan funciones específicas como readr (ideal para trabajar con archivos tipo CSV y readxl (clave para manejar archivos xls o xlsx (Excel)).

1. Definir una Ruta Base:

Definir una ruta base es fundamental para garantizar una correcta organización de los datos dentro de un proyecto.

```
# Definir la ruta base donde están los archivos
ruta_base <- "C:/Usuarios/MiUsuario/Documentos/Datos"
```

2. Construir la Ruta del Archivo:

Una vez definida la ruta base, se utiliza la función **file.path()** para construir la ruta completa del archivo a importar.

```
# Construir la ruta para un archivo CSV
ruta_csv <- file.path(ruta_base, "datos_2024.csv")

# Construir la ruta para un archivo Excel
ruta_excel <- file.path(ruta_base, "reporte_ventas.xlsx")
```

Importar Datos desde CSV, Excel y Otras Fuentes

UNICISO
WWW.PORTALUNICISO.COM

3. Cargar el Archivo en R

El formato CSV es ampliamente usado para el análisis de datos. Para cargar el archivo se utiliza la función **read_csv()** del paquete **readr**.

```
library(readr)  # Cargar la librería para leer CSV
datos_csv <- read_csv(ruta_csv)
```

Para definir el delimitador del archivo se utiliza **read_delim**():

Para cargar archivos Excel se utiliza la función **read_excel()** del paquete **readxl**.

```
install.packages("readxl")
library(readxl)  # Cargar la librería para leer archivos Excel
datos_excel <- read_excel(ruta_excel, sheet = 1)  # Leer la primera hoja
```

Adicionalmente, si se quiere cargar una hoja específica dentro del archivo de Excel se utiliza **excel_sheets(),** remplazándolo en la sección del código “**sheets = 1"** por el nombre específico de la hoja.

UNICISO
WWW.PORTALUNICISO.COM

Inspeccionar datos en R

Inspeccionar los datos es un paso fundamental antes de cualquier análisis, ya que permite comprender su estructura y detectar posibles problemas. Es posible identificar valores faltantes, inconsistencias en los tipos de datos y errores en la carga de información, lo que facilita la toma de decisiones sobre las transformaciones necesarias.

Funciones clave para explorar un dataset en R:

Función	Descripción
`head(datos)`	Muestra las primeras filas del dataset.
`tail(datos)`	Muestra las últimas filas del dataset.
`str(datos)`	Muestra la estructura del dataset (tipos de datos y variables).
`summary(datos)`	Resume estadísticamente las variables (mínimo, máximo, media, NA, etc.).
`dim(datos)`	Devuelve el número de filas y columnas.
`colnames(datos)`	Lista los nombres de las columnas.

UNICISO
WWW.PORTALUNICISO.COM

Manejo de tipos de datos en R

Reconocer los **tipos de datos en un dataset es crucial, ya que cada variable posee un formato específico que define cómo puede ser manipulada y analizada**. Un tipo de dato incorrecto puede provocar errores en cálculos, gráficos y transformaciones, afectando la calidad del análisis.

Principales tipos de datos en R:

Tipo de dato	Descripción	Ejemplo	Código
Numérico	Valores continuos o discretos que permiten operaciones matemáticas.	10, 3.14, -5	as.numeric()
Entero (integer)	Números sin decimales, definidos con una L al final.	5L, -12L	as.integer()
Carácter (character)	Cadenas de texto, representadas entre comillas.	"Rstudio", "Economía"	as.character()
Factor	Variables categóricas con niveles predefinidos.	"Bajo", "Medio", "Alto"	as.factor()
Lógico (logical)	Valores booleanos que representan TRUE o FALSE.	TRUE, FALSE	as.logical()
Fecha (Date/POSIXct)	Representación de fechas y horas en R.	"2024-03-18"	as.Date()

Para verificar el tipo de datos en R se utiliza el código: **class("variable"),** siendo "variable" la variable que se quiere analizar.

UNICISO
WWW.PORTALUNICISO.COM

Filtrar y seleccionar datos en R

El filtrado y la selección de datos son procesos esenciales para centrar el análisis en la información relevante. Filtrar “**filter()”** permite extraer filas específicas según condiciones definidas, mientras que seleccionar “**select()”** ayuda a enfocarse solo en las columnas necesarias.

```
filter(datos, columna == valor)
```

Extrae solo las filas donde la columna cumple una condición.

```
select(datos, columna1, columna2)
```

Conserva solo las columnas especificadas.

```
filter(datos, columna %in% c(valor1, valor2))
```

Filtra múltiples valores.

```
select(datos, -columna)
```

Excluye una columna específica.

UNICISO
WWW.PORTALUNICISO.COM

Transformaciones Básicas

Crear nuevas variables con mutate():

Permite crear nuevas columnas sin modificar la estructura del dataset original, lo que facilita la exploración y el análisis sin perder información.

Cálculos matemáticos

Crear nuevas variables con operaciones aritméticas.

Transformaciones

Aplicar funciones sobre columnas existentes.

Condiciones con if_else()

Generar categorías según condiciones.

```
mutate(datos, nueva_columna = columna1 * 2)
```

Crea una nueva columna.

```
mutate(datos, nueva_columna =
case_when(condición1 ~ valor1, condición2 ~ valor2)):
```

Crear categorías.

```
mutate(datos, log_columna = log(columna))
```

Aplica transformación logarítmica.

Uso con case_when()

Crear múltiples categorías en una variable.

Ejemplo de uso de mutate en R

```
# Crear una nueva variable con el precio total (incluye IVA)
datos <- mutate(datos, precio_final = precio * 1.19)
```

Crear una variable a partir de un cálculo matemático.

```
# Clasificar ingresos como "Bajos", "Medios" o "Altos"
datos <- mutate(datos, categoria = case_when(
 ingreso < 1000 ~ "Bajo",
 ingreso >= 1000 & ingreso < 5000 ~ "Medio",
 ingreso >= 5000 ~ "Alto"
))
```

Crear categorías en basandose en tipo de ingresos “Bajos”, “Medios” y “Altos”.

Modificar datos con rename() y recode ():

Funciones como **rename()** y **recode()** facilitan la limpieza y organización de la información, asegurando que los datos sean más comprensibles y útiles para el análisis.

Función	Objetivo	Ejemplo
rename()	Cambiar el nombre de una columna.	Renombrar "ventas_totales" a "total_ventas".
recode()	Modificar valores dentro de una columna.	Cambiar "M" y "F" por "Masculino" y "Femenino".

```
# Renombrar una columna
datos <- rename(datos, total_ventas = ventas_totales)

# Modificar una variable creando una versión en mayúsculas
datos <- mutate(datos, categoria_mayus = toupper(categoria))

# Reemplazar valores en una variable
datos <- mutate(datos, genero = recode(genero, "M" = "Masculino", "F" = "Femenino"))
```

Ordenar datos con arrange():

Permite reordenar filas según una o más columnas, en orden ascendente o descendente.

Función	Objetivo	Ejemplo
arrange(datos, columna)	Ordena de menor a mayor.	Ordenar ventas de menor a mayor.
arrange(datos, desc(columna))	Ordena de mayor a menor.	Ordenar por ingresos más altos primero.
arrange(datos, coll, col2)	Ordena por múltiples criterios.	Primero por región, luego por fecha.

```
# Ordenar un dataset por la columna "ventas" de menor a mayor
datos <- arrange(datos, ventas)

# Ordenar por "ingresos" de mayor a menor
datos <- arrange(datos, desc(ingresos))

# Ordenar por región y, dentro de cada región, por fecha
datos <- arrange(datos, region, fecha)
```

Manejo de valores faltantes con drop_na() y replace_na():

Función	Objetivo	Ejemplo
drop_na()	Elimina filas con valores faltantes en una o más columnas.	Remover registros con datos incompletos.
replace_na()	Sustituye valores NA por un valor específico.	Rellenar valores faltantes con "Desconocido".
is.na()	Identifica valores faltantes en una columna.	Ver cuántos valores `NA` hay en "edad".

Es fundamental **detectar y manejar los valores** faltantes de manera adecuada para evitar sesgos y mejorar la calidad de los resultados.

```
# Eliminar filas con valores faltantes en cualquier columna
datos <- drop_na(datos)

# Eliminar solo si "ventas" tiene valores faltantes
datos <- drop_na(datos, ventas)

# Reemplazar valores NA en la columna "categoria" por "Desconocido"
datos <- datos %>% mutate(categoria = replace_na(categoria, "Desconocido"))
```

UNICISO
WWW.PORTALUNICISO.COM

Transformaciones Avanzadas

Agrupar datos con group_by() y resumir con summarise():

Al combinar **group_by**() con **summarise**(), se pueden calcular estadísticas clave como promedios, sumas o conteos, facilitando la interpretación de la información.

group_by():
Agrupa los datos según una o más variables.

summarise():
Calcula métricas resumen sobre los grupos creados.

n(): Cuenta el número de observaciones por grupo.

```
# Agrupar por categoría y calcular el promedio de ventas
datos_resumen <- datos %>%
  group_by(categoria) %>%
  summarise(promedio_ventas = mean(ventas, na.rm = TRUE))
```

```
# Contar el número de registros por región
datos_resumen <- datos %>%
  group_by(region) %>%
  summarise(total_registros = n())
```

Nota: %>% representa al operador **pipe** útil para concatenar múltiples **dplyr** operaciones.

Unir bases de datos con left_join(), right_join(), full_join()

Función	Objetivo	Ejemplo
left_join()	Mantiene todas las filas de la tabla izquierda y une los datos coincidentes de la tabla derecha.	Unir una tabla de clientes con otra de compras.
right_join()	Mantiene todas las filas de la tabla derecha y une los datos coincidentes de la tabla izquierda.	Obtener todos los productos vendidos, incluso sin clientes registrados.
full_join()	Mantiene todas las filas de ambas tablas, llenando con NA cuando no hay coincidencias.	Combinar registros de ventas de diferentes años.

Unir bases de datos permite **consolidar información y facilitar su análisis al combinar diferentes fuentes** en una sola estructura.

```
# Unir datos de clientes y compras manteniendo todos los clientes
datos_completos <- left_join(clientes, compras, by = "id_cliente")

# Unir datos manteniendo todas las compras, incluso si no hay cliente asociado
datos_completos <- right_join(clientes, compras, by = c("columna1" = "columna2"))
```

Siempre para unir bases se necesita utilizar una llave **“id”** el cual se incluye en el “**by”.**

Pivotear datos con pivot_longer() y pivot_wider()

Los datos pueden almacenarse en formatos amplios (muchas columnas) o largos (más filas con valores categorizados). Dependiendo del análisis, puede ser necesario transformar la estructura de la base de datos para facilitar cálculos y visualizaciones.

pivot_longer():

Convierte columnas en filas, útil cuando hay muchas variables separadas en columnas.

```
# Transformar un dataset amplio a formato largo
datos_largos <- datos %>%
  pivot_longer(cols = c(enero, febrero, marzo),
               names_to = "mes",
               values_to = "ventas")
```

Long Format

Team	Variable	Value
A	Points	88
A	Assists	12
A	Rebounds	22
B	Points	91
B	Assists	17
B	Rebounds	28
C	Points	99
C	Assists	24
C	Rebounds	30
D	Points	94
D	Assists	28
D	Rebounds	31

pivot_wider():

Convierte filas en columnas, útil cuando se quiere una tabla más compacta.

```
# Transformar un dataset largo a formato amplio
datos_anchos <- datos_largos %>%
  pivot_wider(names_from = "mes",
              values_from = "ventas")
```

Wide Format

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

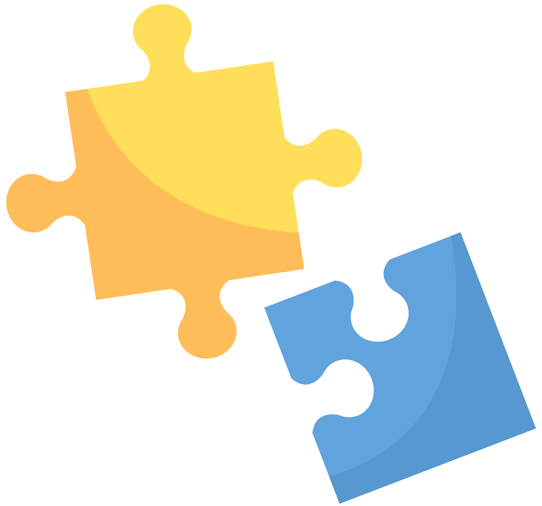
Combinar columnas con unite() y separar información con separate()

Dependiendo del análisis, puede ser necesario unir valores en una columna o dividir una columna en varias para facilitar la manipulación y el análisis.

Función	Objetivo	Ejemplo
unite()	Combina dos o más columnas en una sola.	Unir "nombre" y "apellido" en "nombre_completo".
separate()	Divide una columna en múltiples columnas.	Separar "fecha" en "año", "mes" y "día".

```
# Unir columnas "nombre" y "apellido" en una sola
datos <- datos %>%
  unite(col = "nombre_completo", nombre, apellido, sep = " ")

# Separar una columna de fecha en año, mes y día
datos <- datos %>%
  separate(col = "fecha", into = c("año", "mes", "día"), sep = "-")
```



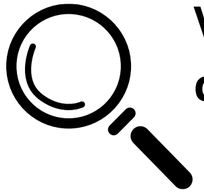
Errores comunes y Mejores Prácticas

Al transformar datos en R, pueden surgir errores que afectan la integridad y precisión del análisis. Identificar y corregir estos problemas es esencial para obtener resultados confiables.

Tipo de error	Descripción	Solución recomendada
Errores de tipos de datos	Conversión incorrecta de factores a numéricos, interpretación errónea de caracteres.	Usar as.character() antes de as.numeric(), validar con str().
Errores en manipulación de Data Frames	Índices incorrectos, nombres de columnas mal referenciados.	Usar colnames() para verificar nombres y select() para evitar errores.
Errores en funciones	Aplicar mean() o sum() a datos no numéricos.	Asegurar que las columnas sean numéricas con is.numeric().
Errores en la fusión de datos	Merge incorrecto por claves erróneas, generación de duplicados.	Verificar claves con unique() antes de usar left_join().
Errores en filtrado de datos	Uso incorrecto de condiciones en filter(), manejo inadecuado de NA.	Revisar con `summary()` y aplicar `na.omit()` o `replace_na()`.

Organización del código: Usar comentarios (#) para describir cada paso.

Nombres bien definidos: Evitar abreviaturas confusas o caracteres especiales.



Verificación constante: Usar str(), summary() y head() para revisar los datos en cada transformación.

Uso de pipes (%>%) Facilita la lectura y evita la creación de múltiples objetos intermedios.

UNICISO
WWW.PORTALUNICISO.COM

Referencias

- AWS. (2024). ¿Qué es ETL? - Explicación de extracción, transformación y carga (ETL). https://aws.amazon.com/es/what-is/etl/
- IBM. (2024, June 19). ¿Qué es la transformación de datos? https://www.ibm.com/es-es/think/topics/data-transformation
- Keilor Rojas-Jimenez. (2022a). Capítulo 5 Transformación, Estandarización e Imputación de Datos. In Ciencia de Datos para Ciencias Naturales. https://bookdown.org/keilor_rojas/CienciaDatos/transformaci%C3%B3n-estandarizaci%C3%B3n-e-imputaci%C3%B3n-de-datos.html
- Keilor Rojas-Jimenez. (2022b). Ciencia de Datos para Ciencias Naturales. https://bookdown.org/keilor_rojas/CienciaDatos/
- r4ds. (2023, June 6). R Para Ciencia de Datos. https://es.r4ds.hadley.nz/
- RCODER. (2025). Transformación de datos en R [Filtrar, Agregar, Ordenar y Resumir]. https://r-coder.com/manipulacion-datos-r/transformacion-datos/
- RPubs. (2024). RPubs—Transformación y estandarización de variables. https://rpubs.com/fer20/semana12_transformacion_y_estandarizacion
- Williams, G. (2011). Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery. Springer. https://doi.org/10.1007/978-1-4419-9890-3

CITA DE LA GUÍA

Ramirez, J.P. (2025). Transformar datos en estadística R. UNICISO.
Disponible en: www.portaluniciso.com

SÍGUENOS:

UNICISO
WWW.PORTALUNICISO.COM