Modulo 4: Manipulación de Datos con dplyr y tidyr en R

02 de July de 2025

Introducción

Introducción al “tidyverse”

- El tidyverse es una colección de paquetes de R diseñados para la ciencia de datos..
- Comparten una filosofía de diseño subyacente, gramática y estructuras de datos, haciendo que la manipulación de datos sea más intuitiva y consistente.

Introducción al "tidyverse"

- Filosofía de Datos "Tidy"
  - La filosofía "tidy" (ordenado) es fundamental en el tidyverse:
  - Cada variable es una columna: Cada tipo de observación que registras debe tener su propia columna.
  - Cada observación (o registro) es una fila: Cada caso o unidad experimental debe ser una fila distinta.
  - Cada unidad de observación debe ser claramente definible: Facilita la comprensión y el análisis.

Introducción al “tidyverse”

- Ventajas:
  - Código más legible (uso de pipes %>%).
  - Consistencia de funciones: mismos nombres en dplyr, ggplot2, tidyr, readr.
  - Pipeline armonizado: lectura.

Operaciones con dplyr

dplyr es un paquete fundamental para la manipulación de datos. Proporciona un conjunto consistente de "verbos" para realizar las operaciones más comunes:

- filter(): seleccionar filas.
- select(): seleccionar columnas.
- mutate(): crear o modificar columnas.
- arrange(): ordenar filas.
- summarise() y group_by(): resumir datos.
- distinct(): obtener filas únicas.
- rename(): renombrar columnas.

Operaciones con dplyr

filter() - Filtrar filas

- Filtra filas según condiciones lógicas.
- Múltiples condiciones se separan por comas (equivalente a "&").
- Se pueden usar funciones lógicas: %in%, between(), is.na().

Operaciones con dplyr

Ejemplos:

data("mtcars") # Usamos el dataset mtcars

Filtrar coches con 6 cilindros y más de 100 caballos de fuerza

```
mtcars_filtrado <- mtcars %>% filter(cyl == 6, hp > 100)
print(head(mtcars_filtrado))
```

Coches con 4 o 6 cilindros

```
mtcars_cyl <- mtcars %>% filter(cyl %in% c(4, 6))
print(head(mtcars_cyl))
```

Coches con un mpg entre 18 y 22

```
mtcars_mpg_range <- mtcars %>% filter(between(mpg, 18, 22))
print(head(mtcars_mpg_range))
```

Operaciones con dplyr

select() - Seleccionar columnas

- Selecciona columnas específicas.
- Permite rangos: select(col1:col5).
- Usar helper functions: starts_with("prefijo"), ends_with("sufijo"), contains("texto"), matches("regex"), everything().

Operaciones con dplyr

Ejemplos:

data("iris") # Usamos el dataset iris

Seleccionar las columnas Sepal.Length y Species

iris_sel <- iris %>% select(Sepal.Length, Species) print(head(iris_sel))

Seleccionar todas las columnas que terminan en ".Width"

iris_width <- iris %>% select(ends_with(".Width"))
print(head(iris_width))

Seleccionar todas las columnas excepto Petal.Length

iris_no_petal_length <- iris %>% select(-Petal.Length)
print(head(iris_no_petal_length))

Operaciones con dplyr

mutate() - Añadir/modificar columnas

- Añade nuevas columnas o modifica existentes.
- case_when() es muy útil para crear variables categóricas basadas en condiciones.

Operaciones con dplyr

Ejemplos:

data(“faithful”) # Usamos el dataset faithful

Crear una nueva columna para el tiempo de espera en minutos y categorizar la duración de la erupción

faithful_mutated <- faithful %>% mutate(waiting_minutes = eruptions * 60, eruption_category = case_when(eruptions < 3 ~ “corta”, eruptions >= 3 & eruptions < 4 ~ “media”, TRUE ~ “larga”))
print(head(faithful_mutated))

Operaciones con dplyr

arrange() - Ordenar filas

- Ordena filas por una o varias columnas.
- Por defecto, el orden es ascendente. Usa desc() para ordenar descendentemente.

Operaciones con dplyr

Ejemplos:

data("iris") # Usamos el dataset iris

Ordenar por Sepal.Length de forma ascendente

```
iris_ord_asc <- iris %>% arrange(Sepal.Length)
print(head(iris_ord_asc))
```

Ordenar por Species (ascendente) y luego por Petal.Width (descendente)

```
iris_ord_multi <- iris %>% arrange(Species, desc(Petal.Width))
print(head(iris_ord_multi))
```

Operaciones con dplyr

summarise() y group_by() - Resumir datos

- Primero agrupar (group_by()), luego resumir (summarise()).
- Usar ungroup() después para eliminar agrupaciones si se va a seguir transformando.
- Otras funciones útiles en summarise(): median(), sd(), min(), max(), n_distinct().

Operaciones con dplyr

Ejemplos:

data(“iris”) # Usamos el dataset iris

Calcular la longitud media y máxima del sépalo por especie

iris_resumen <- iris %>% group_by(Species) %>% summarise(promedio_sepal_length = mean(Sepal.Length, na.rm = TRUE), max_sepal_length = max(Sepal.Length, na.rm = TRUE), num_observaciones = n(), .groups = “drop” # Para desagrupar automáticamente después del summarise) print(iris_resumen)

Operaciones con dplyr

distinct() - Extraer filas únicas

- Extrae filas únicas según columnas especificadas.
- Usa .keep_all = TRUE para mantener todas las columnas de las filas únicas.

Operaciones con dplyr

Ejemplos:

data("CO2") # Usamos el dataset CO2

Obtener una lista de los tipos de plantas únicos

plant_types <- CO2 %>% distinct(Type) print(plant_types)

Obtener filas únicas basadas en la combinación de Plant y Type

unique_plants_full <- CO2 %>% distinct(Plant, Type, .keep_all = TRUE) print(head(unique_plants_full))

Operaciones con dplyr

rename() - Renombrar columnas

- Renombra columnas de un data frame.
- La sintaxis es nuevo_nombre = nombre_antiguo.

Operaciones con dplyr

Ejemplos:

data("mtcars") # Usamos el dataset mtcars

Renombrar 'mpg' a 'millas_por_galon' y 'hp' a 'caballos_fuerza'

mtcars_renombrado <- mtcars %>% rename(millas_por_galon = mpg, caballos_fuerza = hp) print(head(mtcars_renombrado))

Operaciones con dplyr

Encadenamiento con Pipes (%>%)

- El operador pipe %>% (del paquete magrittr, cargado con tidyverse) hace que la salida de la función anterior sea el primer argumento de la siguiente función.
- Facilita la lectura de flujos de trabajo, haciendo el código más secuencial y legible.

Operaciones con dplyr

Ejemplos:

data("quakes") # Usamos el dataset quakes (terremotos)

Filtrar terremotos con profundidad menor a 400km, seleccionar latitud, longitud y magnitud, y ordenar por magnitud descendente

quakes_processed <- quakes %>% filter(depth < 400) %>% select(lat, long, mag) %>% arrange(desc(mag)) print(head(quakes_processed))

Operaciones con dplyr

Pipes: Ejemplo Comparativo

- El pipe permite expresar una secuencia de operaciones de forma más natural.
- El código con pipes es claramente más legible y sigue un flujo lógico.

Con Pipes:

```
data("airquality")
air_filtered_arranged <- airquality %>% filter(Temp > 70) %>%
select(Ozone, Temp, Month) %>% arrange(desc(Temp))
print(head(air_filtered_arranged))
```

Sin Pipes (equivalente):

```
air_filtered_arranged_no_pipe <- arrange( select( filter(airquality, Temp
> 70), Ozone, Temp, Month ), desc(Temp) )
print(head(air_filtered_arranged_no_pipe))
```

Reestructuración con tidyr

- tidyr es un paquete para hacer que los datos estén "tidy" (ordenados), es decir, en un formato que facilita el análisis y la visualización. Sus funciones principales son:
  - pivot_longer() (antes gather()): de ancho a largo.
  - pivot_wider() (antes spread()): de largo a ancho.
  - separate(): dividir una columna en varias.
  - unite(): unir varias columnas en una.
  - Manejo de valores faltantes: drop_na(), replace_na().
  - pivot_longer() (de ancho a largo)
    - Convierte de formato ancho ("wide") a formato largo ("long").
  - Útil cuando los nombres de las columnas son en realidad valores de una variable.

Reestructuración con tidyr

Ejemplos:

Datos de rendimiento académico de estudiantes

rendimiento_wide <- data.frame(Estudiante = c("Ana", "Pedro"), Matematicas_Q1 = c(85, 78), Matematicas_Q2 = c(90, 82), Ciencias_Q1 = c(70, 65), Ciencias_Q2 = c(75, 68)) print("Datos Originales (Wide):") print(rendimiento_wide) rendimiento_long <- rendimiento_wide %>% pivot_longer(cols = starts_with("Matematicas") | starts_with("Ciencias"), names_to = "Asignatura_Trimestre", values_to = "Calificacion") print("Datos Transformados (Long):") print(rendimiento_long)

Ejemplos:

pivot_wider() (de largo a ancho)

- Convierte de formato largo a ancho.
- Útil para crear columnas a partir de valores de una columna existente.

Datos de ventas por vendedor y región (formato largo)

ventas_largo <- data.frame(Vendedor = c("Juan", "Juan", "Maria", "Maria"), Region = c("Norte", "Sur", "Norte", "Sur"), Ventas = c(1000, 1200, 1500, 1300)) print("Datos Originales (Long):") print(ventas_largo) ventas_ancho <- ventas_largo %>% pivot_wider(names_from = Region, values_from = Ventas) print("Datos Transformados (Wide):") print(ventas_ancho)

Ejemplos:

separate() y unite()

- separate(columna, into = c("parte1","parte2"), sep = "_"): divide el contenido de una columna en varias basándose en un separador.
- unite(col_nueva, columnas, sep = "_"): une múltiples columnas en una nueva, con separador.

Ejemplo con separate:

datos_id <- data.frame(id_producto = c("A-123", "B-456", "C-789")) print("Datos Originales (Separate):") print(datos_id) datos_separados <- datos_id %>% separate(id_producto, into = c("categoria", "codigo"), sep = "-") print("Datos Separados:") print(datos_separados)

Ejemplos:

separate() y unite()

- separate(columna, into = c(“parte1”,“parte2”), sep = “_“): divide el contenido de una columna en varias basándose en un separador.
- unite(col_nueva, columnas, sep = “_“): une múltiples columnas en una nueva, con separador.

Ejemplo con unite:

datos_unidos <- datos_separados %>% unite(id_reconstruido, categoria, codigo, sep = “_“) print(”Datos Unidos:“) print(datos_unidos)

Manejo de valores faltantes con tidyr

- drop_na(): elimina filas que contienen NA en las columnas especificadas.
- replace_na(): reemplaza NAs por un valor dado.

Ejemplos:

Manejo de valores faltantes con tidyr

datos_con_na <- data.frame(A = c(1, 2, NA, 4), B = c(NA, 6, 7, 8), C = c(9, 10, 11, NA)) print("Datos con NA:") print(datos_con_na)

Eliminar filas con NA en la columna A

datos_sin_na_A <- datos_con_na %>% drop_na(A) print("Drop NA en columna A:") print(datos_sin_na_A)

Reemplazar NAs en la columna B con 0 y en C con 99

datos_replace_na <- datos_con_na %>% replace_na(list(B = 0, C = 99)) print("Replace NA:") print(datos_replace_na)

Casos Prácticos

Caso Práctico 1: Reestructuración de Datos de Temperaturas

- Supongamos que tenemos un data frame de temperaturas mensuales registradas en diferentes ciudades.
- Queremos analizar estas temperaturas en un formato largo para facilitar cálculos o visualizaciones.

Caso Práctico 2: Agrupación y Resumen de Datos de Vuelos

- Partiendo del conjunto de datos nycflights13::flights, queremos extraer el mes y el número total de vuelos por aerolínea.
- Esto nos permitirá ver la actividad de cada aerolínea a lo largo del año.

¡Gracias!

¿Alguna pregunta sobre la manipulación de datos con dplyr y tidyr?