



Preparación y Análisis de datos

Capítulo 1: Manipulación avanzada de datos

Mg. J. Eduardo Gamboa U.

2025-03-24

Presentación

Docente: Mg. Jesús Eduardo Gamboa Unsihuay

Correo: jgamboa@lamolina.edu.pe

Repositorio: <https://github.com/jeguns/Preparacion-y-analisis-de-datos-con-R>

Descripción

Este curso está diseñado para personas con conocimientos básicos en R que desean profundizar en la manipulación de datos, el análisis inferencial y la generación de reportes automatizados. A través de un enfoque práctico, los participantes aprenderán a preprocesar datos de manera eficiente, aplicar técnicas estadísticas para la toma de decisiones y comunicar sus hallazgos de manera efectiva mediante reportes reproducibles en RMarkdown

Contenido

1. Manipulación avanzada de datos:

- ▶ Manejo de datos con el paquete dplyr
- ▶ Transformación de estructuras de datos con el paquete tidyr
- ▶ Manejo de fechas con el paquete lubridate
- ▶ Manipulación de datos textuales con el paquete stringr

2. Análisis inferencial de datos:

- ▶ Intervalos de confianza con R
- ▶ Pruebas de hipótesis con R
- ▶ Regresión lineal con R

3. Generación de reportes con RMarkdown

Manejo de datos con el paquete dplyr

dplyr es un paquete de R diseñado por Hadley Wickham (RStudio) para la manipulación eficiente de datos. Es parte del ecosistema tidyverse.

Proporciona una “gramática” (particularmente verbos) para la manipulación y operaciones con data frames.

Principales funciones de dplyr:

- ▶ `select()`: Selecciona columnas específicas.
- ▶ `filter()`: Filtra filas según condiciones.
- ▶ `mutate()`: Crea o modifica columnas.
- ▶ `arrange()`: Ordena filas según valores de una o más columnas.
- ▶ `group_by()`: Agrupa los datos por categorías.
- ▶ `left_join()`, `inner_join()`, `right_join()`, `full_join()` : Une tablas de datos.

Operador pipe

Operador pipe nativo: `|>`

- ▶ Disponible desde la versión 4.1.0 de R
- ▶ No requiere cargar ningún paquete
- ▶ Siempre es obligatorio el uso de paréntesis

Operador pipe de `magrittr`: `%>%`

- ▶ Disponible con el paquete `magrittr` (o `dplyr` o `tidyverse`)
- ▶ Cuando la función solo tiene un argumento, es opcional el uso de paréntesis

Ejemplo

```
x = c(1,5,6,3,-2)
sum(x)
```

```
[1] 13
```

```
x |> sum()
```

```
[1] 13
```

```
library(dplyr)
x %>% sum
```

```
[1] 13
```

Función select()

```
df |> select(posiciones o nombres de las variables a ser seleccionadas)
```

```
df |> select(-posiciones o nombres de las variables a ser eliminadas)
```

Ejemplo

Leer el archivo INEI.csv.

```
df = read.csv2('INEI.csv')
```

Seleccionar las columnas Departamento, Provincia y Cantidad.

```
df |> select('Departamento','Provincia','Cantidad')
```

```
df |> select(3, 4, 8)
```

Seleccionar todas las columnas excepto ubigeo_reniec y ubigeo_inei

```
df |> select('-ubigeo_reniec', '-ubigeo_inei')
```

```
df |> select(-1, -2)
```


Función pull()

```
df |> pull(posición o nombre de la variable a ser seleccionada)
```

Ejemplo

Seleccionar la columna Distrito como un vector

```
df |> pull(Distrito)
```

```
df |> pull(4)
```

Función filter()

```
df |> filter(reglas de filtro)
```

Ejemplo

Filtrar los registros correspondientes al sexo masculino

```
df |> filter(Sexo == "M")
```

Filtrar los registros correspondientes a cantidades inferiores a 100 habitantes

```
df |> filter(Cantidad < 100)
```

Filtrar los registros correspondientes al sexo masculino **y** a cantidades inferiores a 100 habitantes

```
df |> filter((Sexo == "M") & (Cantidad < 100))
```

Filtrar los registros correspondientes al sexo masculino **o** a cantidades inferiores a 100 habitantes

```
df |> filter((Sexo == "M") | (Cantidad < 100))
```

Función mutate()

```
df |> mutate(nueva_variable = regla para crear la nueva variable)
df |> mutate(variable_existente = regla para modificar la variable)
```

Ejemplo

Convertir el sexo a minúscula, almacenándolo en la misma variable.

```
df |> mutate(Sexo = tolower(Sexo))
```

Crear una variable que se llame Alta_Cantidad si la cantidad de habitantes es mayor a 300.

```
df |> mutate(Alta_Cantidad = ifelse(Cantidad>300,"Sí","No"))
```

Crear una variable binaria (0=No, 1=Sí) para distinguir al grupo vulnerable de adultos mayores (60 a más años)

```
Edades_am = c("60-64", "65-69", "70-74", "75-79", "80+")
df |> mutate(Es_AdultoMayor = ifelse(Edad_Anio %in% Edades_am, 1, 0))
```

¿Cómo hacer los 3 cambios anteriores en simultáneo?

```
Edades_am = c("60-64", "65-69", "70-74", "75-79", "80+")
df |> mutate(Sexo          = tolower(Sexo),
             Alta_Cantidad = ifelse(Cantidad>300,"Sí","No"),
             Es_AdultoMayor = ifelse(Edad_Anio %in% Edades_am, 1, 0))
```

De otra manera:

```
Edades_am = c("60-64", "65-69", "70-74", "75-79", "80+")
df |>
  mutate(Sexo          = tolower(Sexo)) |>
  mutate(Alta_Cantidad = ifelse(Cantidad>300,"Sí","No")) |>
  mutate(Es_AdultoMayor = ifelse(Edad_Anio %in% Edades_am, 1, 0))
```

Función arrange()

```
df |> arrange(variable) # orden alfabético A - Z, o de menor a mayor  
df |> arrange(desc(variable)) # orden alfabético Z - A, o de mayor a menor  
df |> arrange(-variable numérica) # orden de mayor a menor
```

Ejemplo

Ordenar el data frame según la cantidad de habitantes, de menor a mayor

```
df |> arrange(Cantidad)
```

Ordenar el data frame según la cantidad de habitantes, de mayor a menor

```
df |> arrange(-Cantidad)  
df |> arrange(desc(Cantidad))
```

Ordenar alfabéticamente según departamento, de la A a la Z

```
df |> arrange(Departamento)
```

Ordenar alfabéticamente según departamento, de la Z a la A

```
df |> arrange(desc(Departamento))
```

Ordenar alfabéticamente según departamento (de la A a la Z) y luego según provincia (de la A a la Z)

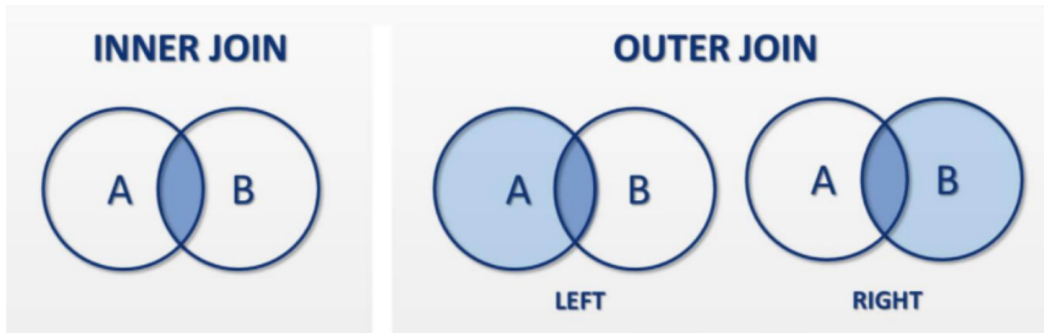
```
df |> arrange(Departamento,Provincia)
```

Ordenar alfabéticamente según departamento (de la A a la Z) y luego según provincia (de la Z a la A)

```
df |> arrange(Departamento,desc(Provincia))
```

Funciones de integración de datos

En ocasiones, nuestros datos no están presentes en un único data frame y necesitamos juntar dos o más de estos.



- ▶ `inner_join(x, y)`: Retorna las filas de `x` que tienen valores coincidentes en `y`, manteniendo todas las columnas de ambas tablas. Solo conserva las coincidencias.
- ▶ `left_join(x, y)`: Retorna todas las filas de `x` y agrega las columnas de `y` cuando hay coincidencias. Si no existen coincidencias, se completan con NA.
- ▶ `right_join(x, y)`: Retorna todas las filas de `y` y agrega las columnas de `x` cuando hay coincidencias. Si no existen coincidencias, se completan con NA.
- ▶ `full_join(x, y)`: Retorna todas las filas de `x` y `y`, combinando sus columnas. Cuando no hay coincidencias en alguna de las tablas, los valores faltantes se completan con NA.

Ejemplo

Data frame de empleados

```
empleados <- data.frame(  
  ID = c(1, 2, 3, 4),  
  Nombre = c("Luisa", "Bruno", "Ana", "Renato"),  
  Departamento = c("Ventas", "TI", "RRHH", "Marketing")  
)
```

Data frame de salarios

```
salarios <- data.frame(  
  ID = c(2, 3, 5),  
  Salario = c(5000, 6000, 4500)  
)
```

empleados

	ID	Nombre	Departamento
1	1	Luisa	Ventas
2	2	Bruno	TI
3	3	Ana	RRHH
4	4	Renato	Marketing

salarios

	ID	Salario
1	2	5000
2	3	6000
3	5	4500

```
empleados |> inner_join(salarios, by = "ID")
```

	ID	Nombre	Departamento	Salario
1	2	Bruno	TI	5000
2	3	Ana	RRHH	6000

```
empleados |> left_join(salarios, by = "ID")
```

	ID	Nombre	Departamento	Salario
1	1	Luisa	Ventas	NA
2	2	Bruno	TI	5000
3	3	Ana	RRHH	6000
4	4	Renato	Marketing	NA

```
empleados |> right_join(salarios, by = "ID")
```

	ID	Nombre	Departamento	Salario
1	2	Bruno	TI	5000
2	3	Ana	RRHH	6000
3	5	<NA>	<NA>	4500

```
empleados |> full_join(salarios, by = "ID")
```

	ID	Nombre	Departamento	Salario
1	1	Luisa	Ventas	NA
2	2	Bruno	TI	5000
3	3	Ana	RRHH	6000
4	4	Renato	Marketing	NA
5	5	<NA>	<NA>	4500

Transformación de estructuras de datos con tidyr

tidyr es un paquete de R diseñado por Hadley Wickham (RStudio) para limpiar y transformar datos en un formato ordenado. Es parte del ecosistema tidyverse. Su utilidad en el preprocesamiento es diversa, para tareas como:

- ▶ Transformación entre formatos ancho y largo
- ▶ Separación y unión de columnas de texto
- ▶ Manejo de valores faltantes

Transformación de formato ancho a largo

Función `pivot_longer`

```
df |>
  pivot_longer(
    cols = columnas_a_convertir, # Columnas que pasarán a formato largo
    names_to = "nombre_nueva_columna", # Columna con nombres originales
    values_to = "nombre_valores" # Columna que contendrá los datos
  )
```

Ejemplo

Supongamos que tenemos este data frame de un diseño experimental

```
x1 = c(15,14,18,17)
x2 = c(20,19,15,17)
x3 = c(12,13,11,16)
datos = data.frame(trat1 = x1, trat2 = x2, trat3 = x3)
datos
```

	trat1	trat2	trat3
1	15	20	12
2	14	19	13
3	18	15	11
4	17	17	16


```
library(tidyr)
datos |> pivot_longer(cols = c("trat1","trat2","trat3"),
                      names_to = "Tratamiento",
                      values_to = "Rendimiento")
```

```
# A tibble: 12 x 2
```

	Tratamiento	Rendimiento
	<chr>	<dbl>
1	trat1	15
2	trat2	20
3	trat3	12
4	trat1	14
5	trat2	19
6	trat3	13
7	trat1	18
8	trat2	15
9	trat3	11
10	trat1	17
11	trat2	17
12	trat3	16

Transformación de formato largo a ancho

Función `pivot_wider`

```
df |>
  pivot_wider(
    names_from = nombre_columna_a_convertir,
    # Columna que originará columnas

    values_from = nombre_columna_valores
    # Columna con los valores que se distribuirán en las columnas
  )
```

Ejemplo

Supongamos que tenemos este data frame:

```
x1 = c("Ingreso","Egreso","Ingreso","Egreso","Ingreso","Egreso")
x2 = c("Lunes","Lunes","Martes","Martes","Miércoles","Miércoles")
x3 = c(20,16,25,12,30,18)
datos = data.frame(tipo = x1, dia = x2, cantidad = x3)
datos
```

	tipo	dia	cantidad
1	Ingreso	Lunes	20
2	Egreso	Lunes	16
3	Ingreso	Martes	25
4	Egreso	Martes	12
5	Ingreso	Miércoles	30
6	Egreso	Miércoles	18

```
datos |> pivot_wider(names_from = dia,  
                     values_from = cantidad)
```

```
# A tibble: 2 x 4
```

	tipo	Lunes	Martes	Miércoles
	<chr>	<dbl>	<dbl>	<dbl>
1	Ingreso	20	25	30
2	Egreso	16	12	18

Separación de columnas de texto

Función separate

```
df |>
  separate(col = columna_a_expandir, # columna que se expandirá
           into = nuevos_nombres, # nuevas columnas que se crearán
           sep = separador) # caracter que servirá para separar
```

Ejemplo

[illegible]

```
df
```

```
      NombreCompleto  
1      Ana Juárez Ramírez  
2      Raúl Gómez Vargas  
3 Beatriz Castillo Castillo
```

```
df |>
```

```
  separate(NombreCompleto, into = c("Nombre", "Apellido"), sep = " ")
```

```
      Nombre Apellido  
1      Ana   Juárez  
2      Raúl   Gómez  
3 Beatriz Castillo
```

Unión de columnas de texto

Función unite

```
df |>
  unite(col = nueva_columna, # Nombre de la nueva columna combinada
        columnas_a_unir,    # Columnas que se unirán
        sep = separador)    # Caracter que se usará para unir los valores
```

Ejemplo

```
x1 = c("EP","ZT","CC")
x2 = c("03","02","01")
x3 = c("01","02","01")
x4 = c(3,4,2,3,3,4)
df = data.frame(codfac = x1, nivel = x2, numero = x3, creditos = 4)
```

```
df
```

	codfac	nivel	numero	creditos
1	EP	03	01	4
2	ZT	02	02	4
3	CC	01	01	4

```
df |> unite(col = cod_curso,  
           codfac, nivel, numero,  
           sep = "")
```

	cod_curso	creditos
1	EP0301	4
2	ZT0202	4
3	CC0101	4

Eliminación de valores faltantes

Función `drop_na`

```
df |> drop_na()
```

Ejemplo

```
x1 = c("María","Ana","Fernando","Eduardo")
x2 = c(2,1,NA,4)
x3 = c("Ate","Lince","Bellavista",NA)
df = data.frame(Persona = x1, Mascotas = x2, Distrito = x3)
```

```
df
```

	Persona	Mascotas	Distrito
1	María	2	Ate
2	Ana	1	Lince
3	Fernando	NA	Bellavista
4	Eduardo	4	<NA>

```
df |> drop_na()
```

	Persona	Mascotas	Distrito
1	María	2	Ate
2	Ana	1	Lince

Susbtitución de valores faltantes

Función `replace_na`

```
df |> replace_na(list(variable1 = valor_a_sustituir,  
                      variable2 = valor_a_sustituir))
```

Ejemplo

```
x1 = c("María","Ana","Fernando","Eduardo")  
x2 = c(2,1,NA,4)  
x3 = c("Ate","Lince","Bellavista",NA)  
df = data.frame(Persona = x1, Mascotas = x2, Distrito = x3)
```

```
df
```

	Persona	Mascotas	Distrito
1	María	2	Ate
2	Ana	1	Lince
3	Fernando	NA	Bellavista
4	Eduardo	4	<NA>

```
df |> replace_na(list(Mascotas = 0, Distrito = "Sin datos"))
```

	Persona	Mascotas	Distrito
1	María	2	Ate
2	Ana	1	Lince
3	Fernando	0	Bellavista
4	Eduardo	4	Sin datos

Manejo de fechas con el paquete lubridate

El paquete lubridate facilita el trabajo con fechas en R, simplificando su creación, conversión, manipulación y extracción de componentes.

Creación de fechas a partir de strings

Las funciones ymd(), dmy(), mdy(), etc., convierten texto en fechas reales

```
library(lubridate)
```

```
fecha1_str = "2025-03-26"
```

```
fecha1_str |> str()
```

```
chr "2025-03-26"
```

```
fecha1_date = fecha1_str |> ymd()
```

```
fecha1_date |> str()
```

```
Date[1:1], format: "2025-03-26"
```

```
fecha2_str  = "07/Apr/2025"  
fecha2_str |> str()
```

```
chr "07/Apr/2025"
```

```
fecha2_date = fecha2_str |> dmy()  
fecha2_date |> str()
```

```
Date[1:1], format: "2025-04-07"
```

```
fecha3_str = "2025-03-18 14:30:45"  
fecha3_str |> str()
```

```
chr "2025-03-18 14:30:45"
```

```
fecha3_date = fecha3_str |> ymd_hms()  
fecha3_date |> str()
```

```
POSIXct[1:1], format: "2025-03-18 14:30:45"
```

Recordemos el uso de la función mutate:

Considere el siguiente data frame:

```
x1 = c("Mauricio","Katy","Pablo","Ana")
x2 = c(19980512,20051230,20070707,20010919)
df = data.frame(Nombre = x1, FechaNac = x2)
df
```

	Nombre	FechaNac
1	Mauricio	19980512
2	Katy	20051230
3	Pablo	20070707
4	Ana	20010919

Convertir la columna FechaNac a formato fecha.

Extracción de componentes de una fecha

```
fecha = ymd("2025-03-01")
```

```
fecha |> year()
```

```
[1] 2025
```

```
fecha |> month()
```

```
[1] 3
```

```
fecha |> day()
```

```
[1] 1
```

```
fecha = ymd("2025-03-01")  
fecha |> yday()
```

```
[1] 60
```

```
fecha |> wday(label = TRUE, abbr = FALSE)
```

```
[1] sábado
```

```
7 Levels: domingo < lunes < martes < miércoles < jueves < ... < sábado
```

```
fecha |> semester()
```

```
[1] 1
```

```
fecha |> quarter()
```

```
[1] 1
```

```
fecha |> week()
```

```
[1] 9
```

```
fecha = ymd("2025-03-01")  
fecha |> year()
```

```
[1] 2025
```

```
fecha |> year() <- 2024  
fecha
```

```
[1] "2024-03-01"
```

```
fecha_hora = ymd_hms("2025-03-18 14:30:45")  
fecha_hora |> hour()
```

```
[1] 14
```

```
fecha_hora |> minute()
```

```
[1] 30
```

```
fecha_hora |> second()
```

```
[1] 45
```

Recordemos el uso de la función mutate:

Considere el siguiente data frame:

```
x1 = c("Mauricio","Katy","Pablo","Ana")
x2 = c(19980512,20051230,20070707,20010919)
df = data.frame(Nombre = x1, FechaNac = x2)
df
```

	Nombre	FechaNac
1	Mauricio	19980512
2	Katy	20051230
3	Pablo	20070707
4	Ana	20010919

Añadir tres columnas: una para el año de nacimiento, otra para el semestre y una tercera para el día de la semana en que nacieron las personas.

Adición o sustracción de periodos de tiempo

```
fecha = ymd("2025-03-01")  
fecha1 = fecha + days(5)  
fecha1
```

```
[1] "2025-03-06"
```

```
fecha2 = fecha - weeks(2)  
fecha2
```

```
[1] "2025-02-15"
```

```
fecha3 = fecha + years(4)  
fecha3
```

```
[1] "2029-03-01"
```

```
fecha_hora = ymd_hms("2025-03-18 14:30:45")  
fecha_hora + hours(3)
```

```
[1] "2025-03-18 17:30:45 UTC"
```

```
fecha_hora - minutes(10)
```

```
[1] "2025-03-18 14:20:45 UTC"
```

```
fecha_hora - seconds(44)
```

```
[1] "2025-03-18 14:30:01 UTC"
```

Manipulación de datos textuales con el paquete stringr

`stringr` es un paquete de R diseñado por Hadley Wickham (RStudio) que contiene herramientas internamente coherentes para trabajar con cadenas de caracteres, es decir, secuencias de caracteres entre comillas. Facilita tareas como extracción, transformación, limpieza, búsqueda y análisis de texto, evitando la complejidad de las expresiones regulares.

Carguemos el paquete:

```
library(stringr)
```


Longitud de texto

Función `str_length`

Mide la longitud de una cadena de caracteres

```
cadena |> str_length()
```

Ejemplos

```
m = "miércoles"  
m |> str_length()
```

```
[1] 9
```

```
x = c("Economía", "    Administración y    finanzas ", "Economía ",  
      "Negocios", "Estadística", "Física")  
x |> str_length()
```

```
[1]  8 32  9  8 11  6
```

Función `str_trim`

Recorta espacios en blanco al inicio, al final o en ambos lados.

```
cadena |> str_trim(side)
```

Ejemplos

```
m = "miércoles"
```

```
m |> str_trim()
```

```
[1] "miércoles"
```

```
x = c("Economía", "    Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística", "Física")
```

```
x |> str_trim(side = "both")
```

```
[1] "Economía"           "Administración y finanzas"  
[3] "Economía"           "Negocios"  
[5] "Estadística"        "Física"
```

Función str_squish

Recorta espacios inclusive dentro del texto

```
cadena |> str_squish()
```

Ejemplos

```
m = "miércoles"  
m |> str_squish()
```

```
[1] "miércoles"
```

```
x = c("Economía", "      Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")  
x |> str_squish()
```

```
[1] "Economía"           "Administración y finanzas"  
[3] "Economía"           "Negocios"  
[5] "Estadística"        "Física"
```

Modificación de cadenas

Función `str_to_lower`

Convierte el texto a minúsculas

```
cadena |> str_to_lower()
```

Ejemplos

```
m = "miércoles"  
m |> str_to_lower()
```

```
[1] "miércoles"
```

```
x = c("Economía", "      Administración y      finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")  
x |> str_to_lower()
```

```
[1] "economía"           "      administración y      finanzas "  
[3] "economía "          "negocios"  
[5] "estadística"        "física"
```

Función str_to_upper

Convierte el texto a mayúsculas

```
cadena |> str_to_upper()
```

Ejemplos

```
m = "miércoles"
```

```
m |> str_to_upper()
```

```
[1] "MIÉRCOLES"
```

```
x = c("Economía", "      Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística", "Física")
```

```
x |> str_to_upper()
```

```
[1] "ECONOMÍA"
```

```
"      ADMINISTRACIÓN Y FINANZAS "
```

```
[3] "ECONOMÍA "
```

```
"NEGOCIOS"
```

```
[5] "ESTADÍSTICA"
```

```
"FÍSICA"
```

Función `str_to_title`

Convierte el texto a formato título (cada palabra con mayúscula inicial)

```
cadena |> str_to_title()
```

Ejemplos

```
m = "miércoles"
```

```
m |> str_to_title()
```

```
[1] "Miércoles"
```

```
x = c("Economía", "      Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística", "Física")
```

```
x |> str_to_title()
```

```
[1] "Economía"
```

```
"      Administración Y Finanzas "
```

```
[3] "Economía "
```

```
"Negocios"
```

```
[5] "Estadística"
```

```
"Física"
```

Función str_replace

Reemplaza la primera coincidencia en cada cadena de un vector

```
vector |> str_replace(patron,reemplazo)
```

```
x = c("Economía", "    Administración y    finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")  
x |> str_replace("    finanzas", " negocios internacionales")
```

```
[1] "Economía"  
[2] "    Administración y negocios internacionales "  
[3] "Economía "  
[4] "Negocios"  
[5] "Estadística"  
[6] "Física"
```

Función `str_replace_all`

Reemplaza todas las coincidencias en cada cadena de un vector

```
vector |> str_replace_all(patron,reemplazo)
```

```
x = c("Economía", "      Administración y      finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")  
x |> str_replace_all(" ","_")
```

[1] "Economía"	"____Administración_y____finanzas_"
[3] "Economía_"	"Negocios"
[5] "Estadística"	"Física"

Detección de coincidencias

Función `str_detect`

Detecta la presencia (TRUE) o ausencia (FALSE) de un patrón en un vector de caracteres.

```
cadena |> str_detect(patron)
```

Ejemplos

```
m = "miércoles"  
m |> str_detect("c")
```

```
[1] TRUE
```

```
x = c("Economía", "      Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")  
x |> str_detect("Economía")
```

```
[1] TRUE FALSE TRUE FALSE FALSE FALSE
```

```
x |> str_detect("Ambiental")
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE
```

Función str_which

Encuentra la posición dentro una vector de caracteres en el que se encuentra el patrón de búsqueda.

```
vector |> str_which(patron)
```

Ejemplo

```
x = c("Economía", "      Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")
```

```
x |> str_which("Economía")
```

```
[1] 1 3
```

```
x |> str_which("Ambiental")
```

```
integer(0)
```

Función str_count

Cuenta el número de coincidencias de un patrón en cada elemento de un vector de caracteres

```
vector |> str_count(patron)
```

Ejemplos

```
m = "miércoles"  
m |> str_count("e")
```

```
[1] 1
```

```
x = c("Economía", "      Administración y finanzas ", "Economía ",  
      "Negocios", "Estadística","Física")  
x |> str_count("o")
```

```
[1] 2 0 2 2 0 0
```

```
x |> str_count("n")
```

```
[1] 1 4 1 0 0 0
```

Aplicación 1

1. Una tienda en línea tiene una base de datos con información de ventas y clientes. Sin embargo, los nombres de los productos presentan inconsistencias en mayúsculas/minúsculas y espacios en blanco, y se necesita extraer información clave de las fechas de venta.
 - a. Leer los archivos `ventas.csv` y `clientes.xlsx`, en dos data frames llamados `ventas` y `clientes`
 - b. Estandarizar los nombres de los productos eliminando espacios adicionales y asegurando que la primera letra de cada palabra sea mayúscula.
 - c. Convertir la columna `Fecha` en formato `Date` y extraer el año, mes y día de la semana.
 - d. Usar `left_join()` para unir ambas bases de datos.
 - e. Guardar el data frame procesado en un archivo `ventas_procesadas.csv`.

Aplicación 2

En un curso universitario, los docentes llevan un registro de asistencia en una tabla donde cada columna representa un día de clases. Sin embargo, la base de datos presenta nombres con espacios innecesarios y las fechas están en formato de texto. Además, se necesita calcular cuántas faltas tiene cada estudiante.

- a. Leer el archivo `asistencia.xlsx` en un data frame llamado `asistencia`
- b. Eliminar espacios extra y asegurar que los nombres estén correctamente capitalizados (uso correcto de mayúsculas y minúsculas).
- c. Usar `pivot_longer()` para convertir la tabla, de modo que cada fila represente un estudiante en una fecha específica.
- d. Transformar la columna `Fecha` en formato `Date`
- e. Mostrar el registro de todas las inasistencias.

Aplicación 3

Una tienda recopila reseñas de clientes sobre sus productos. Para mejorar la atención al cliente, quieren identificar reseñas con comentarios negativos y contar cuántas veces se menciona la palabra “producto” en cada comentario.

- a. Leer el archivo reseñas.xlsx en un data frame de nombre reseñas
- b. Crear una nueva columna Menciona_Excelente que contenga TRUE si el comentario incluye la palabra “excelente”.
- c. Crear una nueva columna Menciona_Mala que contenga TRUE si el comentario incluye “mala”, “lento” o “no volveré”.
- d. Crear una nueva columna Cantidad_Producto que cuente cuántas veces aparece la palabra “producto” en cada comentario.
- e. Crear un nuevo data frame con solo los comentarios problemáticos