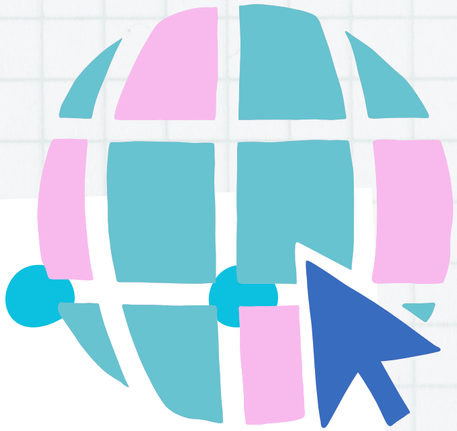


Prácticas para importar datos desde CSV y Excel y exportar resultados en R

DRA. OLGA LÓPEZ FORTIZ

Objetivos y Panorama General

Importar Datos

Entender cómo importar datos desde archivos CSV y Excel en R utilizando funciones como read.csv() y read_excel(). La manipulación de datos es una habilidad esencial en ciencia de datos.

Exportar Resultados

Aprender a exportar datos en diferentes formatos (CSV, Excel, TXT) usando funciones como write.csv() y write.xlsx(). Elegir el formato correcto facilita el análisis y la colaboración.

Práctica en Grupo

Realizar ejercicios prácticos importando y exportando archivos reales. Comentar resultados en grupo, compartir hallazgos y resolver dudas para consolidar el aprendizaje.

Cómo Importar Datos desde un Archivo CSV en R

Paso 1: Usar la función read.csv() para cargar datos tabulares desde un archivo CSV.
Paso 2: Especificar la ruta del archivo y opciones básicas:
• header = TRUE → primera fila como nombres de columna
• sep = "," → separador de campos
Ejemplo de código:
read.csv("datos/ejemplo.csv", header = TRUE, sep = ",")
Verificar la estructura del dataframe con str() o head()

Importar Datos desde Excel en R

Paso 1: Instalar y cargar el paquete readxl
install.packages("readxl") library(readxl)
Paso 2: Usar read_excel() para cargar datos
datos_excel <- read_excel("datos/ejemplo.xlsx", sheet = 1)
Verifica el contenido rápidamente:
head(datos_excel)
💡 Tip: Para múltiples hojas, usa el argumento sheet = "NombreHoja" o sheet = 2.
Puedes listar todas las hojas con excel_sheets("archivo.xlsx").

Exportar Resultados en Diferentes Formatos con R

Exportar a CSV

Función: write.csv()
Ejemplo:
write.csv(resultados,
"resultados.csv",
row.names = FALSE)
Ideal para compatibilidad universal y análisis en hojas de cálculo.

Exportar a Excel

Función: write.xlsx()
Paquetes: openxlsx o writexl
Ejemplo:
write.xlsx(resultados,
"resultados.xlsx")
Perfecto para compartir con usuarios de Excel con formato enriquecido.

Exportar a TXT

Función: write.table()
Ejemplo:
write.table(resultados,
"resultados.txt",
sep = "\t")
Útil para sistemas que requieren texto plano o separadores personalizados.

Ejercicios Prácticos

Ejercicio 1

Importa el archivo "ventas.csv" en R usando read.csv(). Muestra las primeras 6 filas con head(ventas, 6). Verifica la estructura del dataframe con str(). Anota cuántas columnas y filas contiene el archivo.

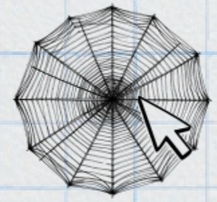
Ejercicio 2

Importa la hoja 2 del archivo "datos_clientes.xlsx" usando read_excel() con el argumento sheet = 2. Muestra el contenido con head(). ¿Cuántas variables tiene? Anota tus observaciones y dudas.

Ejercicio 3

Exporta el dataframe creado a formato CSV usando write.csv(). Luego expórtalo a Excel con write.xlsx(). Verifica que los archivos se guardaron correctamente. Comparte tus resultados con el grupo.

Ejemplo Avanzado: Opciones de read.csv()



```
# Importar CSV con opciones avanzadas
datos <- read.csv(
  "datos/encuesta.csv",
  header = TRUE,
  sep = ";",           # separador punto y coma
  na.strings = c("", "NA", "N/A"),
  stringsAsFactors = FALSE,
  encoding = "UTF-8"
)
str(datos) # verificar estructura
```

Explicación de los argumentos:

sep = ";" -> Archivos europeos usan punto y coma como separador

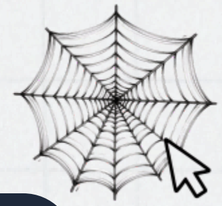
na.strings -> Define qué valores se tratan como NA (datos faltantes)

stringsAsFactors = FALSE -> Evita convertir texto a factores

encoding = "UTF-8" -> Maneja correctamente acentos y caracteres especiales

💡 Tip: Usa file.choose() para abrir un explorador de archivos interactivo en lugar de escribir la ruta manualmente.

Ejemplo: Importar Múltiples Archivos



```
# Importar todos los CSV de una carpeta
archivos <- list.files(
  path = "datos/",
  pattern = "\.csv$",
  full.names = TRUE
)

# Leer cada archivo y combinarlos
lista_datos <- lapply(archivos, read.csv)
```

Funciones clave:

list.files() -> Lista archivos que coinciden con un patrón

lapply() -> Aplica una función a cada elemento de una lista

do.call(rbind, ...) -> Combina múltiples dataframes por filas

full.names = TRUE -> Devuelve la ruta completa del archivo

💡 Tip: Usa purrr::map_dfr(archivos, read.csv) del tidyverse para una solución más elegante.

Ejemplo: Exportar con Formato Personalizado

```
# Exportar a CSV sin nombres de fila
write.csv(datos, "resultados.csv", row.names = FALSE)

# Exportar a Excel con formato
library(openxlsx)
wb <- createWorkbook()
addWorksheet(wb, "Resultados")
writeData(wb, "Resultados", datos)
saveWorkbook(wb, "resultados.xlsx", overwrite = TRUE)

# Exportar a TXT con tabulador
write.table(datos, "resultados.txt", sep = "\t", row.names = FALSE)
```

Opciones importantes:

row.names = FALSE -> No incluir numeros de fila en el archivo

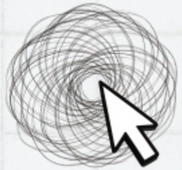
overwrite = TRUE -> Sobrescribir si el archivo ya existe

sep = "\t" -> Usar tabulador como separador (archivos .txt)

fileEncoding = "UTF-8" -> Garantizar compatibilidad con acentos

💡 Tip: Con openxlsx puedes agregar estilos, colores y formatos condicionales a tus hojas de Excel.

Ejercicios Adicionales



Ejercicio 4: Limpieza al importar

1. Importa "productos.csv" con sep = ";"
2. Usa na.strings = c("", "NA", "-") para manejar valores faltantes
3. Verifica con summary() cuantos NA hay por columna
4. Filtra las filas completas con na.omit()
5. Muestra el resultado con head()

Ejercicio 5: Exportar resumen estadistico

1. Importa "ventas.csv" con read.csv()
2. Calcula media, mediana y desviacion con summary()
3. Crea un dataframe con los resultados
4. Exporta a Excel con write.xlsx()
5. Exporta tambien a CSV y compara los archivos

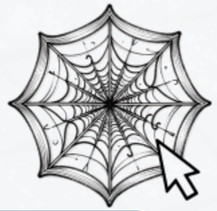
Ejercicio 6: Multiples hojas Excel

1. Usa excel_sheets() para listar las hojas de "informe.xlsx"
2. Importa cada hoja con un loop o lapply()
3. Combina todas las hojas en un solo dataframe
4. Agrega una columna indicando el origen
5. Exporta el resultado final a CSV

Ejercicio 7: Automatizar importacion

1. Crea una carpeta con 3 archivos CSV diferentes
2. Usa list.files() para obtener sus rutas
3. Importa todos con lapply() y read.csv()
4. Combina con do.call(rbind, ...)
5. Exporta el dataframe combinado a Excel
6. Comenta en grupo: ventajas de automatizar

Tabla Resumen de Funciones



Funcion	Paquete	Uso
`read.csv()`	base R	Importar archivos CSV
`read.csv2()`	base R	Importar CSV con separador ; (formato europeo)
`read_excel()`	readxl	Importar archivos Excel (.xls y .xlsx)
`excel_sheets()`	readxl	Listar nombres de hojas en un archivo Excel
`read.table()`	base R	Importar archivos de texto con separador personalizado
`write.csv()`	base R	Exportar dataframe a archivo CSV
`write.xlsx()`	openxlsx	Exportar dataframe a archivo Excel con formato
`write.table()`	base R	Exportar a archivo de texto con separador personalizado
`list.files()`	base R	Listar archivos en una carpeta segun un patron
`lapply()`	base R	Aplicar una funcion a cada elemento de una lista
`do.call(rbind, ...)`	base R	Combinar multiples dataframes por filas
`str()`	base R	Ver estructura del dataframe (tipos de columnas)

Errores Comunes y Soluciones

`Error: cannot open file`

Causa: Ruta incorrecta o archivo no existe

Solucion: Verifica la ruta con file.exists(). Usa getwd() para ver tu directorio actual. Usa rutas relativas o setwd().

`Error: no applicable method for 'read_excel'`

Causa: Paquete readxl no cargado

Solucion: Ejecuta library(readxl) antes de usar read_excel(). Si no esta instalado: install.packages("readxl").

`Columnas con nombres incorrectos`

Causa: Encabezados mal leidos o con espacios

Solucion: Usa check.names = FALSE o col_names en read_excel(). Renombra con colnames(datos) <- c("col1", "col2").

`Datos con caracteres raros (acentos)`

Causa: Problema de codificacion (encoding)

Solucion: Usa fileEncoding = "UTF-8" o "latin1" en read.csv(). Prueba con readr::read_csv() que detecta encoding automaticamente.

`write.csv agrega columna X extra`

Causa: row.names = TRUE por defecto

Solucion: Siempre usa write.csv(datos, "archivo.csv", row.names = FALSE) para evitar la columna de indices.

`Error: object 'datos' not found`

Causa: Variable no creada o nombre incorrecto

Solucion: Verifica que asignaste el resultado: datos <- read.csv(...). Revisa mayusculas y minusculas en el nombre.