Técnicas y Herramientas Modernas II

2024-08-16

Milagros Bravo
Abril Wursten
Lucía Gallart
Juan Ferrero
Lucía López Ballester
Constanza Di Rocco
Lucas Guinea Ritta
Nicolás Guntsche

Grupo Tayloristas

Universidad Nacional de Cuyo
Ciudad de Mendoza

Clase SQLite

SQLite en R

Para trabajar con SQLite en R, primero necesitamos instalar la biblioteca `RSQLite`. Esto se puede hacer desde **Tools > Install Packages** en RStudio, luego seleccionamos `RSQLite` y lo instalamos. También se requiere `DBI` para la conexión y `tidyverse` para la manipulación de datos.

Dentro de los bloques de código, si ponemos `echo=FALSE`, el código no aparecerá en el documento generado. Esto es útil para ocultar comandos auxiliares.

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

CUIDADO: Si al ejecutar el comando library aparece un mensaje de error indicando que la biblioteca no está instalada, debes ir a **Tools > Install Packages...** en RStudio, escribir el nombre de la biblioteca (por ejemplo, RSQLite o tidyverse), y hacer clic en Install. Esto descargará e instalará la biblioteca correspondiente.

Para ejecutar los comandos de programación en la "zona gris" (bloques de código), coloca el cursor al final de la línea que deseas ejecutar y presiona Ctrl + Enter (o Cmd + Enter en Mac). Esto ejecutará el código línea por línea en la consola de R.

Abrir una base de datos de una tabla sola

En este ejemplo, abriremos una conexión a una base de datos de fútbol llamada football-data.db. Mostraremos las tablas que contiene:

```
library("RSQLite")

# Abrir una conexión
con <- dbConnect(drv=RSQLite::SQLite(),dbname="football-data.db")
tables <- dbListTables(con)
tables
```

```
## [1] "campeones_apertura"     "england_premier_league" "spain_la_liga"
## [4] "sqlite_sequence"
```

Ejecutar una Consulta SQL (QUERY)

Ejecutaremos una consulta para ver los partidos donde los equipos visitantes son “Sevilla” o “Real Madrid”.

```
SELECT
 HomeTeam, AwayTeam, FTAG, FTAG
FROM
  spain_la_liga
WHERE
  AwayTeam IN ('Sevilla', 'Real Madrid')
```

Table 1: Displaying records 1 - 10

HomeTeam	AwayTeam	FTAG	FTAG
Ath Bilbao	Real Madrid	2	2
Almeria	Real Madrid	3	3
Alaves	Sevilla	3	3
Celta	Real Madrid	1	1
Osasuna	Sevilla	0	0
Ath Madrid	Real Madrid	1	1
Barcelona	Sevilla	0	0
Girona	Real Madrid	3	3
Sevilla	Real Madrid	1	1
Barcelona	Real Madrid	2	2

Para ejecutar comandos SQL en R Markdown, seleccionamos SQL en lugar de R en el tipo de bloque de código.

```
select * from spain_la_liga
```

Div	Date / Time / HomeTeam / AwayTeam / FTHG	FTAG	FTR	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR	B365H … (odds columns)
SP1	[illegible]	2	A	0	2	A	19	10	5	4	16	25	8	3	1	3	0	0	[illegible]
SP1	[illegible] Valencia	2	A	0	0	D	9	10	3	3	15	17	8	1	3	2	1	0	[illegible]
SP1	[illegible]	[illegible]	D	1	0	H	15	6	2	1	22	9	4	1	2	5	0	0	[illegible]
SP1	[illegible] Las Palmas / Mallorca	1	D	1	0	H	5	12	2	2	11	21	3	6	3	2	0	0	[illegible]
SP1	[illegible] Ath Bilbao / Real Madrid	2	A	0	2	A	7	14	1	8	18	6	7	5	3	1	0	0	[illegible]
SP1	[illegible] Osasuna	2	A	0	1	A	11	11	1	5	11	15	7	2	1	1	0	0	[illegible]
SP1	[illegible] Betis	2	A	0	1	A	18	12	5	5	13	17	9	4	1	3	0	0	[illegible]
SP1	[illegible] Getafe / Barcelona	0	D	0	0	D	5	14	2	4	20	11	3	4	8	2	1	1	[illegible]
SP1	[illegible] Alaves	0	H	1	0	H	12	10	3	2	19	15	2	4	3	2	1	0	[illegible]
SP1	[illegible] Granada / Ath Madrid	1	H	1	0	H	14	7	6	4	15	9	3	3	3	1	0	0	[illegible]

Este bloque de código mostrará todos los registros (filas) y todas las columnas de la tabla `spain_la_liga` en el documento generado por RMarkdown. Esto es útil para visualizar todo el contenido de la tabla directamente en el reporte o documento sin necesidad de exportarlo a un archivo aparte.

Tarea 1

Crear una tabla de alumnos en la misma base:

Creamos una tabla llamada alumnos en la misma base de datos con la siguiente estructura:

```
CREATE TABLE alumnos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre TEXT NOT NULL,
    calificacion INTEGER,
    promedio REAL,
    telefono INTEGER,
    email TEXT DEFAULT "@gmail.com"

);
```

Insertamos algunos registros en la tabla alumnos:

```
INSERT INTO alumnos (id, nombre, calificacion, promedio, telefono, email)
VALUES
(1, "Milagros Bravo", 9 , 7.8 , 2614183531, "milibravo2911"),
(2, "Abril Wursten", 9 , 9.8 , 2622666961, "abrilwursten99"),
```

```
(3, "Constanza Di Rocco", 8 , 9.3 , 2616390959, "cdiroccoarmitano"),
(4, "Lucia Gallart", 8 , 7.3 , 2615414285, "luciagallart"),
(5, "Lucia Lopez", 7 , 8.3 , 2613428978, "lucilopez"),
(6, "Juan Ferrero", 9 , 8.3 , 2616360833, "juanmarcosferreromartinez")
;
```

Verificamos el contenido de la tabla alumnos:

```
select * from alumnos;
```

Table 3: 6 records

id	nombre	calificacion	promedio	telefono	email
1	Milagros Bravo	9	7.8	2614183531	milibravo2911
2	Abril Wursten	9	9.8	2622666961	abrilwursten99
3	Constanza Di Rocco	8	9.3	2616390959	cdiroccoarmitano
4	Lucia Gallart	8	7.3	2615414285	luciagallart
5	Lucia Lopez	7	8.3	2613428978	lucilopez
6	Juan Ferrero	9	8.3	2616360833	juanmarcosferreromartinez

Para eliminar la tabla alumnos:

```
DROP TABLE alumnos;
```

Ejemplo y apertura de la conexión

Código completo en R que crea la conexión a la base de datos, genera las tablas 'population' y 'who' con datos de ejemplo, y luego ejecuta la consulta SQL para fusionar y filtrar los datos según tus criterios.

```
# Cargar las librerías necesarias
library(DBI)
library(RSQLite)
```

```
# Crear la conexión a la base de datos en memoria
conexion <- dbConnect(drv = RSQLite::SQLite(), dbname = ":memory:")
```

```
# Verificar si las tablas existen y eliminarlas si es necesario
if ("population" %in% dbListTables(conexion)) {
  dbRemoveTable(conexion, "population")
}
if ("who" %in% dbListTables(conexion)) {
  dbRemoveTable(conexion, "who")
}
```

```
# Crear ejemplos de los datasets population y who
population <- data.frame(
  country = c("Brazil", "Germany", "Brazil", "Germany"),
  year = c(2000, 2000, 2001, 2001),
  population = c(175000000, 82000000, 176000000, 83000000)
```

```
)

who <- data.frame(
  country = c("Brazil", "Germany", "Brazil", "Germany"),
  year = c(2000, 2000, 2001, 2001),
  new_sp_m3544 = c(1000, 500, 1200, 600)
)
```

```
# Almacenar las tablas en la base de datos
dbWriteTable(conn = conexion, name = "population", value = population)
dbWriteTable(conn = conexion, name = "who", value = who)
```

```
# Ejecutar la consulta SQL para fusionar y filtrar datos
query <- "
SELECT
  who.country, who.year, who.new_sp_m3544, population.population
FROM
  who
LEFT JOIN
  population ON population.country = who.country AND population.year = who.year
WHERE
  who.country IN ('Brazil', 'Germany') AND
  who.year >= 2000 AND
  who.year <= 2010
"
```

```
# Obtener los resultados de la consulta
M1_results <- dbGetQuery(conexion, query)
```

```
# Mostrar las primeras filas del resultado
print(head(M1_results))
```

```
##   country year new_sp_m3544 population
## 1  Brazil 2000         1000   1.75e+08
## 2 Germany 2000          500   8.20e+07
## 3  Brazil 2001         1200   1.76e+08
## 4 Germany 2001          600   8.30e+07
```

```
# Cerrar la conexión
dbDisconnect(conexion)
```