

Acceso a bases de datos: RODBC

Sitio: [Plataforma de Formación On line del Instituto Andaluz de Administración Pública](#)
Curso: (I22F-PT05) Entorno de Programación R
Libro: Acceso a bases de datos: RODBC

Imprimido por: ALFONSO LUIS MONTEJO RAEZ
Día: lunes, 18 de abril de 2022, 10:13

Tabla de contenidos

1. Introducción

- 1.1. Acceso ODBC
- 1.2. Cambio de versión de R

2. Instalación de RODB

3. Trabajando con RODB

- 3.1. Establecer conexión
- 3.2. Ver contenido
- 3.3. Extraer información
- 3.4. Cerrar conexión
- 3.5. Otras funciones

1. Introducción

El paquete **RODBC** emplea la conectividad ODBC (Open Database Connectivity) para establecer la conexión con una base de datos e interactuar con ella mediante SQL.

Mediante este paquete se puede acceder a datos en formatos:

- Access (mdb y accdb).
- Excel (xls yxlsx).
- dBase.
- Oracle.

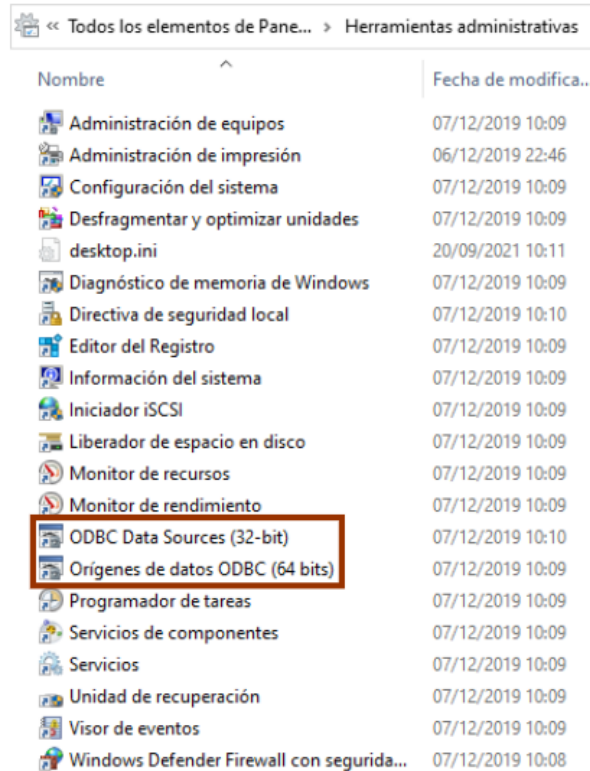
Para que este paquete funcione es necesario tener instalado un acceso ODBC en nuestro ordenador para cada una de las fuentes con las que queramos trabajar.

1.1. Acceso ODBC

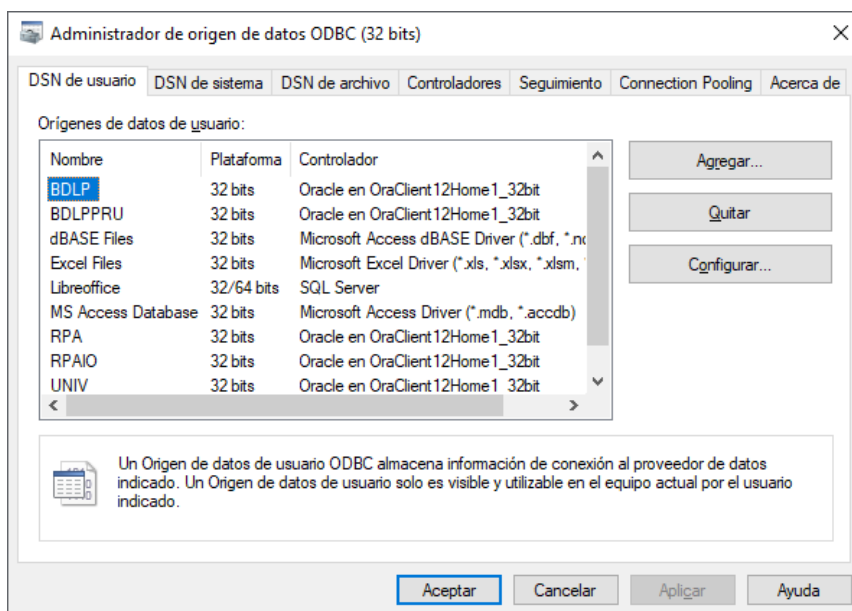
En el caso de ordenadores con sistema operativo Windows, para poder comprobar si tenemos el acceso ODBC instalado y con qué fuentes podemos trabajar, tenemos que ir a la siguiente ruta:

Inicio → Panel de Control → Herramientas administrativas → Orígenes de datos ODBC

Normalmente se tienen instaladas dos versiones: una de 32 bits y otra de 64 bits.



Para que el paquete **RODBC** funcione correctamente en R tenemos que hacerlo desde la versión de 32 bits.



Aquí tenemos disponibles todas las conexiones que tenemos instaladas en nuestro ordenador. Como podemos ver, en mi ordenador, tengo instalados accesos a ficheros Excel, Access, Libreoffice y Oracle.

Normalmente estas conexiones se instalan automáticamente al instalar los programas de Microsoft Office pero, si quisiéramos instalar una nosotros, lo podemos hacer desde el botón **Agregar**.

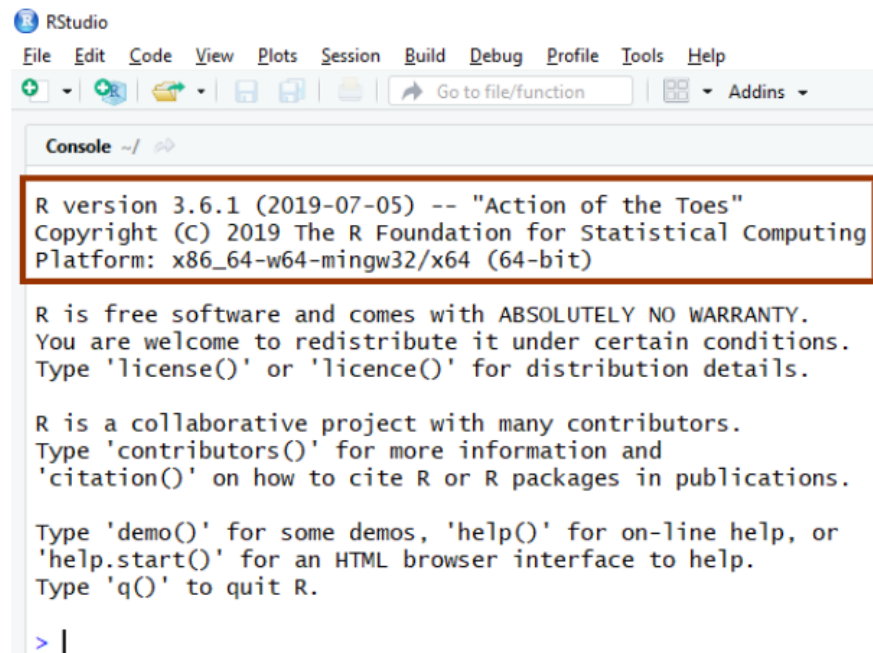
1.2. Cambio de versión de R

Como hemos comentado, para que el paquete RODBC funcione correctamente, tenemos que trabajar con R desde la versión de 32 bits.

Como normalmente trabajaremos con al versión de 64 bits, tenemos que cambiar de versión cada vez que queramos usar este paquete.

Desde RStudio es muy fácil cambiar la versión de R con la que trabajamos. Vamos a verlo.

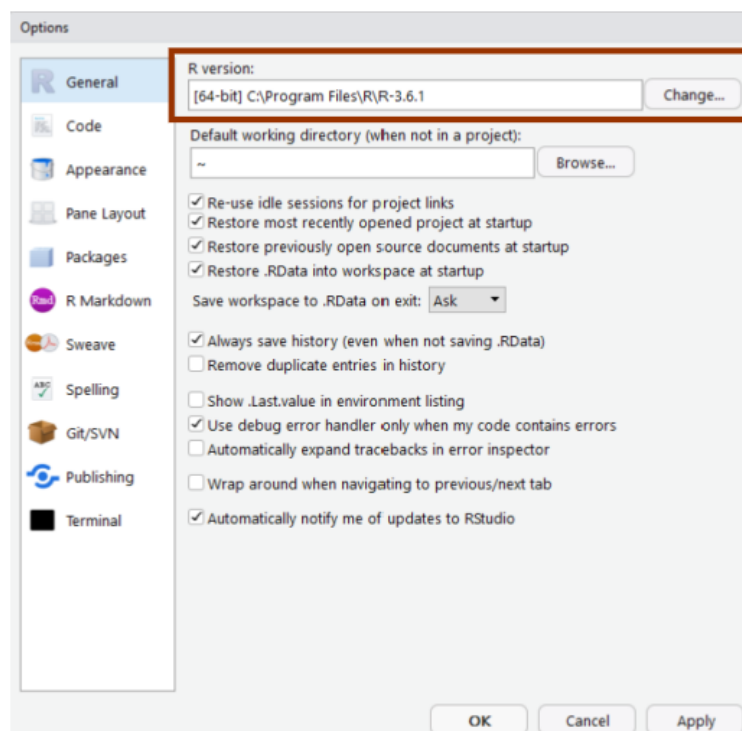
Cuando arrancamos RStudio, en la consola podemos ver la versión de R con la que estamos trabajando.



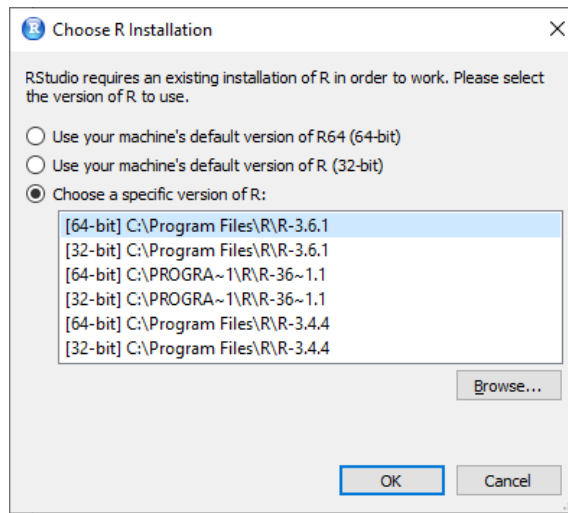
Como podéis ver yo trabajo con la versión de R 3.6.1 en 64 bit.

Para cambiarla tenemos que ir a la siguiente ruta:

Tools → Global Options → R General

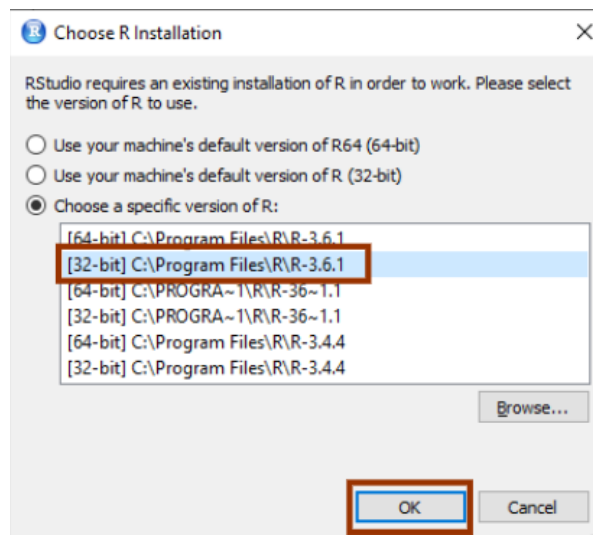


Pulsando el botón **Change...** podemos ver todas las versiones de R que tenemos instaladas.

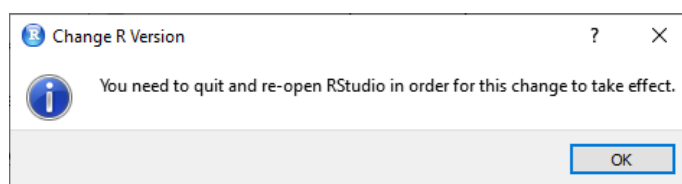


Por defecto aparece marcada con la que estamos trabajando, pero podemos elegir la que queramos. Yo tengo instaladas en mi ordenador dos versiones de R (la 3.6.1 y la 3.4.4) y las correspondientes de 32 y 64 bits para cada una de ellas.

Se trata de marcar con la que queremos trabajar y pulsar el botón **OK**. En nuestro caso elegimos la 3.6.1 de 32 bit.





Al hacerlo nos aparece un mensaje pidiendo que cerremos RStudio y que lo abramos de nuevo para que los cambios surtan efecto.



Pulsamos **OK** y cerramos RStudio.

Al volver a abrirlo, ya nos aparece que estamos trabajando la versión de R que hemos elegido.

 RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
Console ~/ 

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```


2. Instalación de RODBC

RODBC es un paquete que hay que instalar y cargar para que funcione ya que no viene por defecto en el paquete base de R.

La instalación se puede hacer de dos formas:

- **mediante órdenes.**
- **mediante asistente.**

Mediante órdenes

Lo haríamos usando **install.packages('RODBC', dependencies=T)** y a continuación usamos la orden **library(RODBC)** para cargar el paquete y poder trabajar con él.

Mediante asistente

Como para cualquier paquete de R, el programa RStudio nos ofrece la posibilidad de instalar y cargar un paquete desde su asistente.

Este proceso ya lo hemos explicado anteriormente en la Unidad 1. Para más detalles ver:

Unidad 1 → Empezar a trabajar con RStudio → 3. Gestión de paquetes → 3.3. Instalación y 3.4. Carga

3. Trabajando con RODB

El paquete **RODB** nos permite establecer conexiones con una base de datos y trabajar con ella desde R.

Podemos extraer datos desde la base de datos hacia R y también podemos escribir desde R información en la base de datos.

Lo que vamos a ver son una colección de órdenes que nos van a permitir realizar ese trabajo.

3.1. Establecer conexión

Lo primero que tenemos que hacer para trabajar con una base de datos es establecer conexión con ella desde R.

Aquí entra en juego el acceso ODBC que hemos comentado anteriormente. Sin ese acceso no podemos comunicarnos con la base de datos.

RODBC tiene definidas una serie de órdenes para conectarnos a las bases de datos más frecuentes. Básicamente lo que hacemos es acceder a la ubicación que contiene el archivo de la base de datos.

- Para archivos `accdb`. `odbcConnectAccess2007(ruta_archivo)`
- Para archivos `mdb`. `odbcConnectAccess(ruta_archivo)`
- Para archivos `xlsx`. `odbcConnectExcel2007(ruta_archivo)`
- Para archivos `xls`. `odbcConnectExcel(ruta_archivo)`
- Para archivos `dbf`. `odbcConnectDbase(ruta_archivo)`

También existe la orden genérica de conexión para otras bases de datos, por ejemplo Oracle, donde podemos definir todos los parámetros de conexión. La orden sería `odbcConnect()`.

En resultado de esta orden la tenemos que asignar a un objeto que almacene los datos de la conexión ya que vamos a estar haciendo referencia constantemente a la conexión.

Vamos a verlo con dos ejemplos.

Ejemplo 1

Vamos a trabajar con el fichero **Datos_territoriales.mdb** que contiene los datos de municipios y provincias de toda España. El archivo está en formato Access anterior a 2007.

Para este formato de base de datos tenemos una orden específica para conectar con ella. Sería la orden `odbcConnectAccess()`.

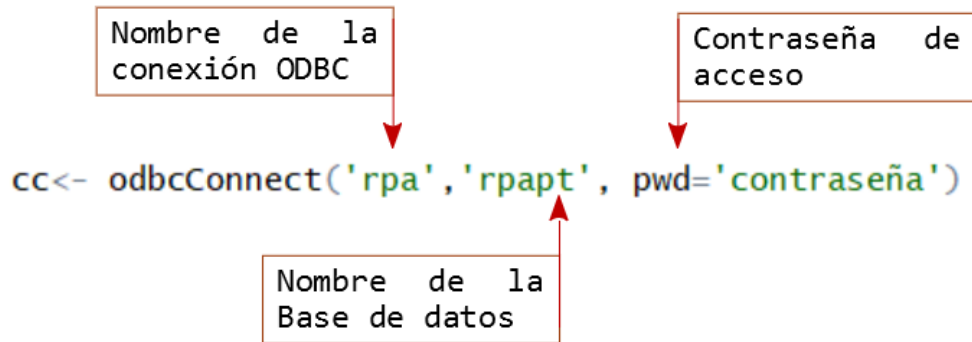
```
> library(RODBC)
> ruta<-'D:\\Curso IAAP\\IAAP I18-PS30\\Temario\\Sesion3\\Datos\\Datos_territoriales.mdb'
> odbcConnectAccess(ruta)->BD
> BD
RODBC Connection 1
Details:
case=nochange
DBQ=D:\\Curso IAAP\\IAAP I18-PS30\\Temario\\Sesion3\\Datos\\Datos_territoriales.mdb
Driver={Microsoft Access Driver (*.mdb)}
DriverId=25
FIL=MS Access
MaxBufferSize=2048
PageTimeout=5
UID=admin
```

Hay que recordar que debemos almacenar el resultado de la conexión en un objeto. En nuestro caso hemos creado **BD** y contiene la conexión a la base de datos.

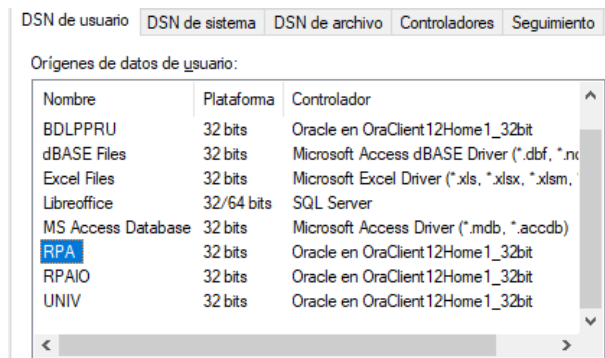
Con esta acción ya tenemos la conexión establecida y podemos empezar a trabajar con la base de datos `Datos_territoriales`.

Ejemplo 2

Vamos a establecer una conexión con una base de datos Oracle llamada `rpapt`. Este tipo, no cuenta con orden propia para acceder a ella por lo que tenemos que usar la de conexión genérica que sería, la orden `odbcConnect()`.



Tenemos que recordar que debe estar definido el acceso ODBC a este tipo de base de datos.



Cuando ejecutamos la orden tenemos creado un objeto llamado **cc** que contiene la información de la conexión.

```
> cc
RODBC Connection 1
Details:
case=nochange
DSN=rpa
UID=rpapt
PWD=*****
DBQ=BDLP2
DBA=W
APA=T
EXC=F
FEN=T
QTO=T
FRC=10
FDL=10
LOB=T
RST=T
```

3.2. Ver contenido

Una vez establecida la conexión con la base de datos, podemos ver el contenido de la misma como pueden ser las tablas.

Con la orden **sqlTables(conexión)** podemos tener un listado de las tablas que contiene la base de datos. Usando esta orden nos aparece un listado con todas las tablas y vistas de la base de datos, incluidas las internas que se usan para el control de la información.

La información básica que obtenemos es: el esquema al que pertenece la tabla (dentro de la base de datos), nombre de la tabla y tipo de tabla.

Este último campo nos permite distinguir entre las tablas de sistema (SYSTEM o SYNONYM) y las que contienen datos (TABLE o VIEW).

También podemos conocer la estructura de una tabla usando la orden **sqlColumns()**.

Con esta orden podemos conocer el nombre de los campos así como su tipo e información sobre su definición.

Ejemplo

Continuando con el archivo de datos territoriales, vamos a ver las tablas que contiene la base de datos.

sqlTables(BD)

Listado de tablas
dentro de **BD**

	TABLE_SCHEM	TABLE_NAME	TABLE_TYPE	REMARKS
1	<NA>	MSysAccessStorage	SYSTEM TABLE	<NA>
2	<NA>	MSysACES	SYSTEM TABLE	<NA>
3	<NA>	MSysIMEXColumns	SYSTEM TABLE	<NA>
4	<NA>	MSysIMEXSpecs	SYSTEM TABLE	<NA>
5	<NA>	MSysNavPaneGroupCategories	SYSTEM TABLE	<NA>
6	<NA>	MSysNavPaneGroups	SYSTEM TABLE	<NA>
7	<NA>	MSysNavPaneGroupToObjects	SYSTEM TABLE	<NA>
8	<NA>	MSysNavPaneObjectIDs	SYSTEM TABLE	<NA>
9	<NA>	MSysObjects	SYSTEM TABLE	<NA>
10	<NA>	MSysQueries	SYSTEM TABLE	<NA>
11	<NA>	MSysRelationships	SYSTEM TABLE	<NA>
12	<NA>	Municipios	TABLE	<NA>
13	<NA>	Provincias	TABLE	<NA>

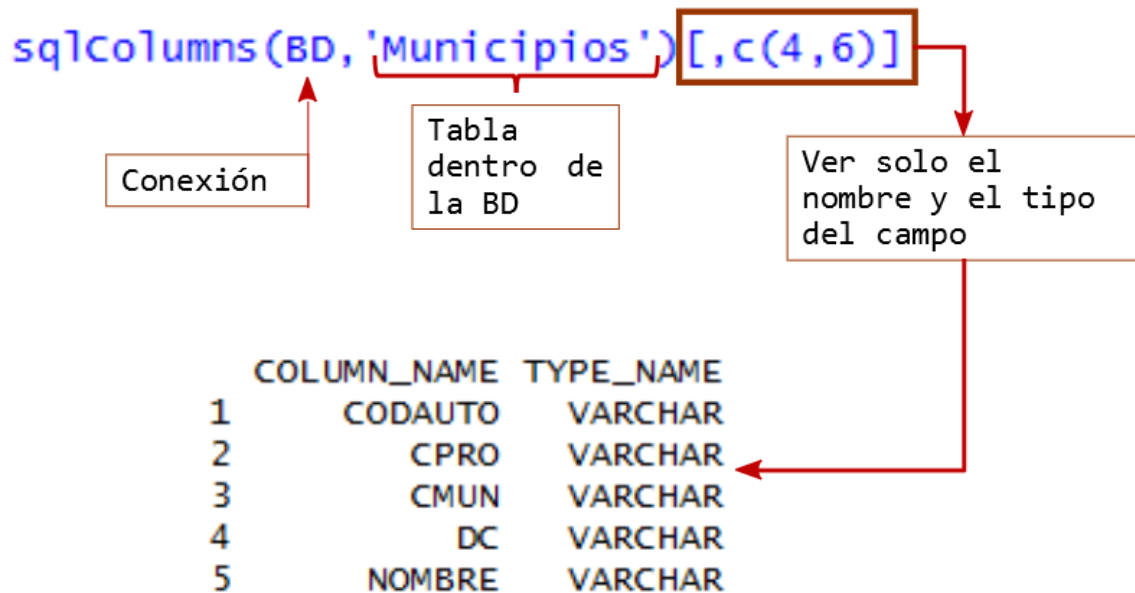
Si usamos la orden **sqlTables** con la conexión **BD** podemos ver las tablas que contiene. Todas las tablas propias del sistema Access, serán las denominadas SYSTEM TABLE y las tablas con los datos que nos interesan son las de tipo TABLE.

De hecho podemos filtrarlas directamente desde la orden.

```
> sqlTables(BD, tableType = "TABLE")$TABLE_NAME  
[1] "Municipios" "Provincias"
```

Las tablas que nos interesan son Municipios y Provincias.

Si queremos saber la información que contiene la tabla Municipios, usamos la orden **sqlColumns()**.



En este caso vemos que la tabla Municipios de la conexión BD, tiene cinco campos de tipo texto (varchar) llamados CODAUTO, CPRO, CMUN, DC y NOMBRE.

3.3. Extraer información

Para acceder al contenido de las tablas de la base de datos se pueden usar las órdenes `sqlFetch()` y `sqlQuery()`.

Con `sqlFetch()` accedemos a todo el contenido de una tabla, mientras que con `sqlQuery()` podemos mandar cualquier consulta en lenguaje SQL a la base de datos y ésta nos devuelve el resultado de la consulta.

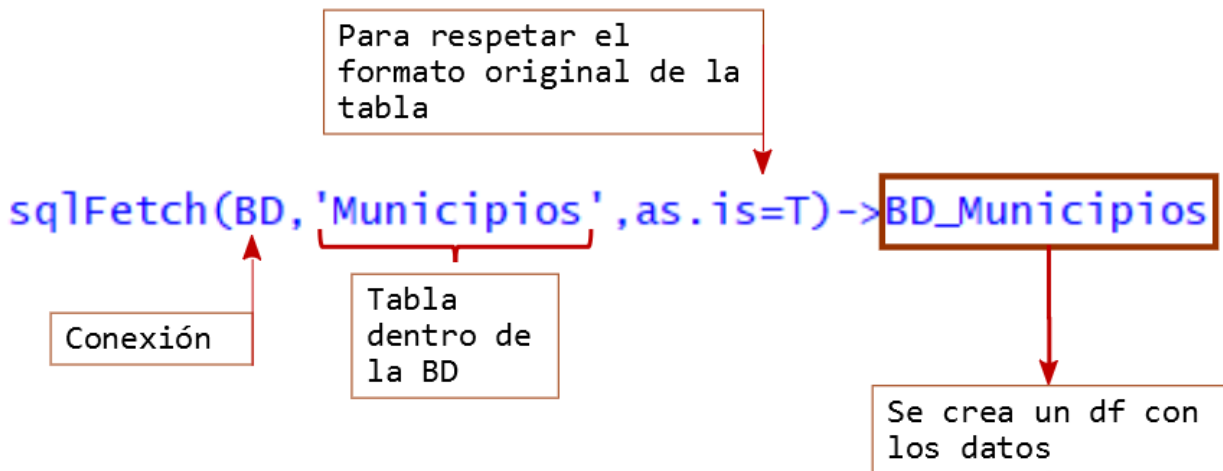
Para mantener el formato original de la tabla en la base de datos, las dos funciones tienen un parámetro llamado `as.is` que debemos poner a `TRUE`.

En los dos casos lo que nos devuelve es un data frame en formato R. Si queremos podemos trabajar con él tenemos que asignar el resultado de la orden a un objeto.

Ejemplo

Continuando con el ejemplo anterior, vamos a llevar a R el contenido de la tabla Municipios de la base de datos.

Para ello usamos la orden `sqlFetch()` indicando la conexión donde se encuentra la tabla y el nombre de la tabla que queremos obtener.



	CODAUTO	CPRO	CMUN	DC	NOMBRE
1	16	01	051	3	Agurain/Salvatierra
2	16	01	001	4	Alegría-Dulantzi
3	16	01	002	9	Amurrio
4	16	01	049	3	Añana
5	16	01	003	5	Aramaio

Para ver la diferencia entre usar el parámetro `as.is` o no hacerlo vamos a ver los dos casos.

```
> sqlFetch(BD, 'Municipios') -> BD_Municipios
> str(BD_Municipios)
'data.frame': 8124 obs. of 5 variables:
 $ CODAUTO: int 16 16 16 16 16 16 16 16 16 ...
 $ CPRO : int 1 1 1 1 1 1 1 1 1 ...
 $ CMUN : int 51 1 2 49 3 6 37 8 4 9 ...
 $ DC : int 3 4 9 3 5 6 6 8 0 1 ...
 $ NOMBRE : Factor w/ 8107 levels "Ababuj", "Abades"
```

Si no lo usamos los valores se convierten en numéricos (perdiendo el valor cero en el primer dígito de los códigos) y los campos de texto se convierten en factores.

```
> sqlFetch(BD,'Municipios',as.is=T)->BD_Municipios
> str(BD_Municipios)
'data.frame': 8124 obs. of 5 variables:
 $ CODAUTO: chr "16" "16" "16" "16" ...
 $ CPRO : chr "01" "01" "01" "01" ...
 $ CMUN : chr "051" "001" "002" "049" ...
 $ DC : chr "3" "4" "9" "3" ...
 $ NOMBRE : chr "Agurain/Salvatierra" "Alegria-Dulantzi"
```

Si lo usamos se mantienen los formatos de texto definidos en la tabla de la base de datos original.

Otra opción para obtener datos de una tabla de la base de datos es la orden **sqlQuery()**.

En este caso se puede pasar una consulta en lenguaje sql para acceder a todos los datos o filtrando la información que queramos. Con esta orden podemos pasar cualquier consulta.

Para obtener todos los datos sería:

```
sqlQuery(BD,'select * from Municipios',as.is=T)
```

Conexión

Consulta en lenguaje SQL

Si queremos filtrar los datos podemos definir la consulta con una cláusula **where**.

```
paste0("select * from Municipios where CODAUTO='',"01'")->consulta
sqlQuery(BD,consulta,as.is=T)->BD_Andalucia
```

	CODAUTO	CPRO	CMUN	DC	NOMBRE
1	01	04	001	0	Abla
2	01	04	002	5	Abrucena
3	01	04	003	1	Adra
4	01	04	004	6	Albanchez
5	01	04	005	9	Alboloduy

Esta consulta filtra los datos de Andalucía

3.4. Cerrar conexión

Siempre que trabajamos con este paquete tenemos que establecer una conexión con la base de datos a través de ODBC.

Cuando terminamos de trabajar con la conexión tenemos que cerrarla para eliminar el acceso y todos datos de la conexión que se quedan en memoria.

Para ello se puede usar la orden **odbcClose()** o **odbcCloseAll()**. Con la primera cerramos la conexión que le indiquemos en el parámetro y con la segunda cerramos todas las conexiones abiertas (sin necesidad de indicar cual).

Ejemplo

Ya hemos terminado de trabajar con la base de datos territoriales y vamos a proceder a cerrar la conexión.

Para ello usamos la orden **odbcClose()** indicando el nombre de la conexión.

```
> odbcClose(BD)

> sqlQuery(BD, consulta, as.is=T)->BD_Andalucia
Error in sqlQuery(BD, consulta, as.is = T) :
  first argument is not an open RODBC channel
```

Una vez cerrada la conexión, si intentamos hacer de nuevo alguna consulta sobre ella nos dará un error al no estar el canal abierto.

3.5. Otras funciones

Además de lo que hemos visto también podemos hacer otras acciones con la base de datos.

Os listo algunas de las más interesantes.

- **sqlSave()**. Crear tablas en la base de datos.
- **sqlUpdate()**. Actualizar una tabla en la base de datos.
- **sqlDrop()**. Borrar una tabla en la base de datos.